



TUGAS AKHIR - KI141502

**IMPLEMENTASI JARINGAN SENSOR NIRKABEL BERBASIS
BLUETOOTH PADA SISTEM PEMANTAUAN KONDISI
KELEMBABAN LAHAN SECARA ADAPTIF**

**REGIN IQBAL MAREZA
NRP 5112100167**

**Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**

**Dosen Pembimbing II
Hudan Studiawan, S.Kom, M.Kom**

**JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
2016**

(Halaman ini sengaja dikosongkan)



UNDERGRADUATE THESES - KI141502

IMPLEMENTATION OF BLUETOOTH BASED WIRELESS SENSOR NETWORK FOR ADAPTIVE LAND MOISTURE CONDITION MONITORING SYSTEM

**REGIN IQBAL MAREZA
NRP 5112100167**

**Supervisor I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**

**Supervisor II
Hudan Studiawan, S.Kom, M.Kom**

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016**

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI JARINGAN SENSOR NIRKABEL BERBASIS BLUETOOTH PADA SISTEM PEMANTAUAN KONDISI KELEMBABAN LAHAN SECARA ADAPTIF

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
REGIN IQBAL MAREZA
NRP : 5112 100 167

Disetujui oleh Dosen Pembimbing Tugas Akhir

1. Waskitho Wibisono, S.Kom., M.Eng.,
NIP: 197410222000031001

2. Hudan Studiawan, S.Kom, M.Kom.
NIP: 198705112012121003



SURABAYA
JUNI, 2016

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI JARINGAN SENSOR NIRKABEL BERBASIS BLUETOOTH PADA SISTEM PEMANTAUAN KONDISI KELEMBABAN LAHAN SECARA ADAPTIF

Nama Mahasiswa : REGIN IQBAL MAREZA
NRP : 5112100167
Jurusan : Teknik Informatika FTIF-ITS
**Dosen Pembimbing 1 : Waskitho Wibisono, S.Kom.,
M.Eng., Ph.D.**
Dosen Pembimbing 2 : Hudan Studiawan, S.Kom, M.Kom.

Abstrak

Kelembaban lahan merupakan salah satu aspek yang penting dalam kehidupan manusia karena memegang peranan di bidang pertanian. Aspek ini merupakan aspek yang perlu dipantau secara konsisten dan terus menerus mengingat kondisi kelembaban tanah yang sewaktu-waktu bisa dengan cepat berubah.

Untuk memantau kondisi yang cepat berubah tersebut, maka dibutuhkan mekanisme pemantauan kondisi kelembaban tanah dari jauh dan real-time, sehingga pihak pengelola lahan langsung dapat mengambil tindakan yang diperlukan.

Penelitian ini menggunakan mikrokontroler arduino, GPS/GPRS/GSM Shield, modul Bluetooth, sensor kelembaban tanah, dan sensor kelembaban udara untuk membuat sebuah jaringan sensor nirkabel berbasis Bluetooth yang dapat digunakan untuk mendapatkan dan mengkalkulasikan data kelembaban tanah dan udara, sehingga kemudian data tersebut dikirim ke server. Metode pengiriman pun diatur sedemikian rupa agar pengiriman dilakukan secara adaptif, yakni mengirimkan data ketika ada perubahan yang signifikan dari kondisi lahan tersebut. Hal ini dilakukan agar mampu menghemat sumber daya

yang digunakan oleh sistem. Untuk menentukan kapan melakukan pengiriman data, sistem menggunakan Fuzzy logic sebagai metode untuk menentukan kapan data akan dikirimkan.

Dari uji coba komponen sensor, didapatkan bahwa sistem yang bekerja secara adaptif, yaitu menyesuaikan dengan kondisi lingkungan implementasinya, bekerja lebih efektif dan efisien. Menghemat konsumsi baterai sebanyak 8,3 persen. Selain itu tingkat keberhasilan pengiriman data hasil pemantauan juga dari uji coba juga 75 persen.

Kata kunci: Jaringan Sensor Nirkabel, Pemantauan Lahan, Arduino, Fuzzy logic, Bluetooth.

IMPLEMENTATION OF BLUETOOTH BASED WIRELESS SENSOR NETWORK FOR ADAPTIVE LAND MOISTURE CONDITION MONITORING SYSTEM

Student's Name : REGIN IQBAL MAREZA
Student's ID : 5112100167
Department : Teknik Informatika FTIF-ITS
First Advisor : Waskitho Wibisono, S.Kom.,
M.Eng., Ph.D.
Second Advisor : Hudan Studiawan, S.Kom,
M.Kom.

Abstract

Soil moisture is one important aspect of human life because it plays one of important roles in agriculture. This aspect is an aspect that needs to be monitored consistently and continuously considering the soil moisture conditions which at times can be changes quickly.

To monitor these rapidly changing conditions, we need remote and real-time soil moisture condition monitoring mechanism, so that the land manager can immediately take the necessary action.

This study uses a microcontroller Arduino, GPS / GPRS / GSM Shield, Bluetooth module, soil moisture sensor, and air humidity sensor to create a wireless sensor networks based on Bluetooth that can be used to obtain and calculate the data of soil moisture and humidity so then the data sent to the server. Delivery method was set up such that the delivery is done adaptively, ie transmit data when there is a significant change from the condition of the land. This is done in order to conserve resources used by the system. To determine when to perform data transmission, using Fuzzy logic system as a method to determine when the data will be sent.

Based on the experiment of sensor component, it was found that the system works adaptively, ie adapting to environmental conditions of implementation, work more effectively and efficiently. Save battery consumption up to 8.3%. In addition to the level of monitoring results also from the trial are also 75%.

Keyword: Wireless Sensor Network, Land Moisture Monitoring, Arduino, Fuzzy logic, Bluetooth.

DAFTAR ISI

LEMBAR PENGESAHAN	v
<i>Abstrak</i>	vii
<i>Abstract</i>	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR GAMBAR	xvii
DAFTAR TABEL	xix
DAFTAR <i>PSEUDOCODE</i>	xxi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Manfaat	3
1.6 Metodologi	3
1.7 Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA	7
2.1 Jaringan Sensor Nirkabel	7
2.2 Mikrokontroler Arduino	9
2.3 Arduino Uno	10
2.4 Arduino Mega 2560	11
2.5 <i>Bluetooth</i>	12
2.6 GPS/GSM/GPRS <i>Shield</i> V3.0	13
2.7 Modul <i>Bluetooth</i> HC-05	14
2.8 Sensor Kelembaban Tanah FC-28	16
2.9 Sensor Suhu dan Kelembaban Udara DHT11	17
2.10 Metode <i>Fuzzy logic</i>	18
2.11 Kelembaban Lahan	20
BAB III PERANCANGAN PERANGKAT LUNAK	24
3.1 Deskripsi Umum Sistem	24
3.2 Arsitektur Umum Sistem	25

3.3	Perancangan Perangkat Keras.....	26
3.3.1	Perancangan pada <i>Slave Node</i>	26
3.3.2	Perancangan pada <i>Master Node</i>	27
3.4	Perancangan Diagram Alir Data <i>Level 0</i>	28
3.5	Perancangan Sifat Adaptif Sistem.....	29
3.6	Perancangan <i>Fuzzy Logic</i>	32
3.6.1	Perancangan <i>Fuzzy Logic</i> pada <i>Slave Node</i>	32
3.6.2	Perancangan <i>Fuzzy Logic</i> pada <i>Master Node</i>	35
3.7	Diagram Alir Aplikasi Sistem.....	36
3.7.1	Diagram Alir Inisialisasi Arduino pada <i>Slave Node</i>	36
3.7.2	Diagram Alir Inisialisasi Arduino pada <i>Master Node</i>	37
3.7.3	Diagram Alir Mendapatkan Nilai Kelembaban Tanah.....	38
3.7.4	Diagram Alir Mendapatkan Nilai Suhu dan Kelembaban Udara.....	39
3.7.5	Diagram Alir Pengiriman Data Menuju <i>Master Node</i> Secara Adaptif.....	40
3.7.6	Diagram Alir Pengiriman Data Menuju <i>Server</i> Secara Adaptif	41
3.8	Rancangan Antarmuka Pengguna	43
BAB IV IMPLEMENTASI.....		45
4.1	Lingkungan Implementasi.....	45
4.1.1	Lingkungan Implementasi Perangkat Keras.....	45
4.1.2	Lingkungan Implementasi Perangkat Lunak	45
4.2	Implementasi Perangkat Keras.....	46
4.2.1	Implementasi Perangkat Keras pada <i>Slave Node</i>	47
4.2.2	Implementasi Perangkat Keras pada <i>Master Node</i>	48
4.3	Implementasi Perangkat Lunak.....	48
4.3.1	Implementasi Perangkat Lunak pada Arduino	49
4.3.2	Implementasi Perangkat Lunak pada Modul <i>Bluetooth</i>	54
BAB V PENGUJIAN DAN EVALUASI		59
5.1	Lingkungan Uji Coba.....	59
5.2	Skenario Uji Coba.....	60
5.2.1	Uji Coba Fungsionalitas	60
5.2.2	Uji Coba Performa.....	68
5.3	Analisis Hasil Uji Coba.....	69

BAB VI KESIMPULAN DAN SARAN.....	72
6.1. Kesimpulan.....	72
6.2. Saran.....	72
DAFTAR PUSTAKA.....	74
LAMPIRAN.....	76
BIODATA PENULIS.....	80

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Gambar 2.1. Ilustrasi WSN [2].....	8
Gambar 2.2. Arduino Uno.....	11
Gambar 2.3. Arduino Mega 2560.....	12
Gambar 2.4. GPS/GSM/GPRS <i>Shield</i> V3.0.....	14
Gambar 2.5. Modul <i>Bluetooth</i> HC-05.....	15
Gambar 2.6. Sensor kelembaban tanah FC-28.....	17
Gambar 2.7. Sensor suhu dan kelembaban udara DHT11.....	18
Gambar 2.8. Cara kerja <i>fuzzy logic</i> secara ringkas.....	19
Gambar 3.1. Rancangan arsitektur umum.....	25
Gambar 3.2. Perancangan perangkat keras.....	26
Gambar 3.3. Perancangan pada <i>slave node</i>	27
Gambar 3.4. Perancangan pada <i>master node</i>	28
Gambar 3.5. Diagram alir data pada <i>level 0</i>	29
Gambar 3.6. Diagram alir berjalannya sistem secara umum.....	30
Gambar 3.7. Gambaran sifat adaptif sistem.....	31
Gambar 3.8. Diagram alir inisialisasi Arduino pada <i>slave node</i>	37
Gambar 3.9. Diagram alir inisialisasi Arduino pada <i>master node</i>	37
Gambar 3.10. Diagram alir mendapatkan nilai kelembaban tanah	39
Gambar 3.11. Diagram alir mendapatkan nilai suhu dan kelembaban udara.....	40
Gambar 3.12. Diagram alir pengiriman data menuju <i>master node</i> secara adaptif.....	41
Gambar 3.13. Diagram alir pengiriman data menuju <i>server</i> secara adaptif.....	42
Gambar 3.14. Rancangan antarmuka pengguna.....	43
Gambar 4.1. Implementasi perangkat keras pada <i>slave node</i>	47
Gambar 4.2. Implementasi perangkat keras pada <i>master node</i> ...	48
Gambar 4.3. Rangkaian HC-05 dalam AT mode.....	55
Gambar 5.1. Sample tanah dalam pengujian.....	60

Gambar 5.2. Respon terhadap AT <i>Command</i>	62
Gambar 5.3. Dua modul <i>Bluetooth</i> yang telah terhubung	63
Gambar 5.4. Uji coba rangkaian <i>slave node</i>	64
Gambar 5.5. Uji coba pada <i>slave node</i>	65
Gambar 5.6. Rangkaian <i>master node</i> dan <i>slave node</i>	66
Gambar 5.7. Uji coba pengiriman data ke <i>master node</i>	67
Gambar 5.8. Hasil uji coba pengiriman ke <i>server</i>	68
Gambar 5.9. Grafik kondisi kelembaban tanah	70
Gambar 5.10. Grafik kondisi kelembaban udara [11]	71

DAFTAR TABEL

Tabel 2.1. Contoh AT <i>Command</i> pada GPS/GPRS/GSM <i>Shield</i> V3.0.....	14
Tabel 2.2. Contoh AT <i>Command</i> pada HC-05	15
Tabel 2.3. Kategori tanah berdasarkan hasil pembacaan sensor ..	20
Tabel 2.4. Kategori kelembaban udara berdasarkan nilai kelembaban relatif	21
Tabel 3.1. Variabel Linguistik Kelembaban Tanah.....	33
Tabel 3.2. Variabel linguistik kelembaban udara	33
Tabel 3.3. Matriks <i>Fuzzy</i> pada <i>slave node</i>	33
Tabel 3.4. <i>Fuzzy rule</i> pada <i>slave node</i>	34
Tabel 3.5. Matriks <i>Fuzzy</i> pada <i>master node</i>	35
Tabel 4.1. Spesifikasi lingkungan implementasi perangkat keras	45
Tabel 4.2. Spesifikasi lingkungan implementasi perangkat lunak	46
Tabel 4.3. Perangkat keras yang dibutuhkan.....	46
Tabel 5.1. Spesifikasi perangkat keras dalam uji coba.....	59
Tabel 5.2. Spesifikasi perangkat lunak dalam uji coba	59
Tabel 5.3. AT <i>Command</i> pada <i>master node</i>	61
Tabel 5.4. Spesifikasi <i>power bank</i>	69

(Halaman ini sengaja dikosongkan)

DAFTAR *PSEUDOCODE*

<i>Pseudocode 4.1. Fungsi setup pada slave node.....</i>	<i>49</i>
<i>Pseudocode 4.2. Menentukan linguistic variables.....</i>	<i>50</i>
<i>Pseudocode 4.3. Menentukan fuzzy rule.....</i>	<i>50</i>
<i>Pseudocode 4.4. Implementasi fungsi loop pada slave node ...</i>	<i>51</i>
<i>Pseudocode 4.5. Fungsi setup pada master node.....</i>	<i>52</i>
<i>Pseudocode 4.6. Inisialisasi fuzzy logic pada master node.....</i>	<i>53</i>
<i>Pseudocode 4.7. Inisialisasi fuzzy rule pada master node.....</i>	<i>53</i>
<i>Pseudocode 4.8. Fungsi SendData</i>	<i>54</i>
<i>Pseudocode 4.9. Fungsi loop pada master node</i>	<i>54</i>
<i>Pseudocode 4.10. Implementasi langkah inisialisasi modul Bluetooth.....</i>	<i>56</i>
<i>Pseudocode 4.11. Implementasi langkah inquiry modul Bluetooth.....</i>	<i>56</i>
<i>Pseudocode 4.12. Implementasi langkah connecting pada modul Bluetooth.....</i>	<i>57</i>

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

1.1 Latar Belakang

Lahan pertanian merupakan sebuah sarana yang berperan penting dalam kehidupan manusia. Maka tentunya dalam hal ini, pengawasan kondisi lahan pertanian adalah satu aktivitas yang vital bagi perawatan lahan itu sendiri. Karena lahan pertanian yang terbuka sangat terpengaruh oleh kondisi dan fenomena alam yang terjadi di sekitarnya. Karena itu dibutuhkan sebuah teknologi tepat guna, efektif dan efisien untuk memantau kondisi lahan pertanian tersebut.

Teknologi jaringan sensor nirkabel adalah salah satu solusi yang bisa ditawarkan untuk mengawasi lahan pertanian. Sensor mampu mengirim data kondisi lahan pertanian secara *real-time* tanpa kita harus awasi terus perkembangannya. Namun, penggunaan sensor ini juga harus diperhatikan masalah efisiensi energinya. Karena itu, konsep jaringan sensor nirkabel ini dipadukan dengan teknologi *Bluetooth* yang dikenal sebagai teknologi nirkabel yang relatif lebih hemat energi.

Sensor yang digunakan dalam penelitian ini adalah sensor yang umum digunakan pada pemantauan lahan yang biasa terdapat pada *greenhouse*, yakni sensor kelembaban udara dan sensor kelembaban tanah. Namun dalam penelitian ini yang digunakan adalah jenis sensor yang lebih sederhana, karena bertujuan untuk membangun sebuah purwarupa dari sistem yang lebih besar. Sensor dalam sistem ini berguna sebagai alat untuk mengumpulkan data yang kelak akan digunakan dalam agregasi data.

Sistem yang dikembangkan dalam penelitian ini adalah bagaimana jaringan sensor nirkabel mengirimkan paket data dengan menggunakan *Bluetooth*. Fokus masalah yang diangkat dalam penelitian ini adalah bagaimana caranya agregasi data bisa dilakukan sedemikian rupa agar pengiriman data bisa dilakukan secara adaptif.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana merancang sebuah jaringan sensor nirkabel berbasis *Bluetooth*?
2. Bagaimana menetapkan situasi dan kondisi untuk melakukan pengiriman data?
3. Bagaimana *slave node* mengirimkan data ke *master node* secara adaptif?
4. Bagaimana *master node* mengirimkan data ke server secara adaptif?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Aplikasi hanya untuk menginformasikan kondisi kelembaban lahan.
2. Perangkat sensor menggunakan mikrokontroler Arduino.
3. Pengiriman data antar *node* menggunakan menggunakan koneksi *Bluetooth*.
4. Aplikasi pemantauan kelembaban lahan ini berbasis web dengan bahasa pemrograman PHP.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini antara lain :

1. Membuat jaringan sensor nirkabel yang efisien dan hemat energi.
2. Membuat sebuah sistem pemantauan kondisi kelembaban lahan yang adaptif.
3. Memantau kondisi kelembaban lahan secara *real-time*.
4. Membuat aplikasi pemantauan kondisi kelembaban lahan yang dapat diakses di mana saja lewat koneksi internet.

1.5 Manfaat

Penelitian ini diharapkan dapat membantu untuk mengelola kondisi lahan dengan mudah dan efektif.

1.6 Metodologi

Metodologi yang digunakan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir
Tahap awal dalam pengerjaan tugas akhir ini adalah penyusunan proposal tugas akhir. Proposal tugas akhir ini berisi deskripsi pendahuluan dari tugas akhir yang akan dibuat. Terdiri dari latar belakang diajukannya usulan tugas akhir, rumusan masalah, batasan masalah, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu dijelaskan pula tinjauan pustaka yang digunakan sebagai referensi pendukung implementasi tugas akhir. Pada proposal ini juga terdapat perencanaan jadwal pengerjaan tugas akhir.
2. Studi literatur
Tahap ini merupakan tahap pengumpulan informasi yang dibutuhkan dalam pengerjaan tugas akhir. Selain itu, di tahap ini pula informasi yang dikumpulkan tersebut dipelajari. Dimulai dari mengumpulkan literatur dan sumber referensi, memilah, kemudian dipahami untuk nantinya dapat berguna dalam pengerjaan tugas akhir.
3. Analisis dan desain perangkat lunak
Tahap ini meliputi analisis dan perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi yang akan diterapkan dalam pengerjaan tugas akhir.
4. Implementasi perangkat lunak
Tahap ini merupakan tahap pengimplementasian rancangan sistem yang telah dibuat. Tahap ini merealisasikan apa yang terdapat pada tahap sebelumnya sehingga kemudian sistem tersebut sesuai dengan apa yang telah direncanakan.
5. Pengujian dan evaluasi

Tahap ini adalah tahapan dimana sistem yang telah dibangun tersebut diuji dengan menggunakan skenario pengujian yang telah dipersiapkan sebelumnya. Tahapan ini bertujuan untuk mengevaluasi sistem serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan tugas akhir yang menjelaskan pengerjaan tugas akhir secara keseluruhan. Mencakup konsep, dasar teori, metode, implementasi, hasil pengerjaan tugas akhir, serta kesimpulan dan saran.

1.7 Sistematika Penulisan

Secara garis besar, buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, batasan, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu terdapat juga metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan mengenai dasar teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi perancangan sistem yang dibuat. Dimulai dari diagram alir yang menjelaskan cara kerja sistem, gambaran arsitektur sistem, dan perancangan antarmuka dari sistem.

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Penjelasan disajikan dalam bentuk *Pseudocode* dari fungsi-fungsi yang digunakan dalam sistem.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian fungsionalitas sistem dengan skenario pengujian yang telah dipersiapkan.

Bab VI Penutup

Bab ini merupakan bab terakhir yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak selanjutnya

(Halaman ini sengaja dikosongkan)

BAB II

TINJAUAN PUSTAKA

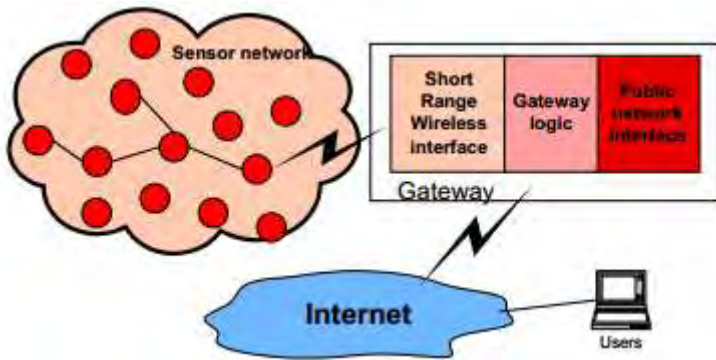
Bab ini berisi penjelasan teori-teori yang berkaitan dengan perancangan alat yang diajukan pada tahap implementasi perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran umum terhadap sistem yang dirancang dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Jaringan Sensor Nirkabel

Jaringan sensor nirkabel atau lebih populer dengan istilah *Wireless Sensor Network* (WSN) dapat didefinisikan secara ringkas sebagai sekumpulan *node* yang terorganisir menjadi sebuah jaringan yang bekerja secara kooperatif [1]. Masing-masing *node* tersebut diartikan sebagai sebuah perangkat kecil (*small device*) yang dilengkapi dengan unit sensor, mikroprosesor, *interface* komunikasi nirkabel, dan sumber tenaga [2].

Cara berkomunikasi dalam WSN ini adalah tersentralisasi, yaitu di mana cara kerja *node-node* tersebut akan terpusat pada sebuah *node*, *node* yang menjadi pusat itulah yang disebut sebagai *gateway* [2]. Dalam penelitian ini, yang berperan sebagai *gateway* adalah *master node*, di mana *master node* akan menjadi pusat dari agregasi data yang terjadi pada jaringan sensor nirkabel. Sedangkan *slave node* berperan sebagai *node-node* yang tersebar di dalam sebuah jaringan WSN.

Beberapa hal penting yang harus diperhatikan dalam pengembangan WSN adalah kolaborasi antar *node* selama pengekseskuan tugas. Masing-masing *node* bertanggung jawab atas perannya masing-masing. *Node-node* sensor yang tersebar pada sebuah jaringan atau *network* bertugas untuk mengumpulkan data untuk kemudian mengirimkan data tersebut ke sebuah *node* yang berperan sebagai *gateway*. *Gateway* itulah yang selanjutnya akan mengirimkan data tersebut ke pengguna atau ke *server*. Untuk ilustrasi bagaimana cara kerja dan peran dari masing-masing *node* dalam WSN dapat dilihat pada Gambar 2.1.



Gambar 2.1. Ilustrasi WSN [2]

Node-node yang tersebar kemudian terhubung dengan *gateway* untuk kemudian membentuk sebuah network. Seperti yang sudah dijelaskan, tugas dari *node-node* tersebut adalah mengumpulkan data, sedangkan *gateway* memiliki tiga peran utama, yaitu untuk *short-range wireless interface*, *gateway logic*, dan *public network interface*.

Short-range wireless interface adalah tentang bagaimana teknologi komunikasi nirkabel itu digunakan. Teknologi yang dipakai dalam penelitian ini adalah teknologi *Bluetooth*. Sehingga *gateway* (yang selanjutnya disebut *master node* sesuai dengan terminologi yang digunakan dalam teknologi *Bluetooth*) yang menginisiasi koneksi di dalam jaringan. *Gateway logic* adalah tentang bagaimana *gateway* mampu mengontrol *node-node* yang terhubung dengannya, sekaligus mengatur data yang masuk. *Public network interface* adalah tentang bagaimana *gateway* berkomunikasi dengan yang ada di luar jaringan, bisa jadi ke pengguna lewat koneksi internet, atau ke sesama jaringan WSN lainnya. [2]

Selain peran dan kolaborasi antar masing-masing *node*, yang harus diperhatikan dalam pengembangan WSN adalah proses pengiriman data. Tidak seperti jaringan sensor lainnya yang langsung mengirimkan data dan menggabungkannya di *server*, pada jaringan sensor nirkabel pada tiap *node*-nya harus bisa

melakukan pemrosesan data mentah. Tujuannya adalah untuk mengurangi jumlah data yang dikirimkan lewat koneksi nirkabel dan melakukan pengiriman data hanya pada data yang relevan saja untuk dikirimkan untuk memudahkan penggabungan data.

Selain itu, yang penting untuk diperhatikan pula adalah masalah konsumsi tenaga. Seperti yang telah dijelaskan sebelumnya mengenai definisi *node* pada WSN, bahwa sebuah *node* adalah sebuah perangkat kecil yang dilengkapi dengan unit sensor, mikroprosesor, *interface* komunikasi nirkabel, dan sumber tenaga. Konsumsi tenaga pada WSN harus diperhatikan mengingat WSN dibangun oleh banyak *node* yang tentu membutuhkan tenaga yang tidak sedikit. Konsumsi tenaga harus diatur sedemikian rupa agar bisa berguna seefisien mungkin. [2]

Topik yang diangkat dalam penelitian ini adalah menyoroti tentang bagaimana data itu diproses dalam sebuah *node* dan keterkaitannya dengan kapan waktu pengiriman itu dilakukan. Pada penelitian ini akan dilakukan implementasi metode *fuzzy logic* sebagai metode pengagregasian dan pemrosesan data. Penjelasan mengenai metode *fuzzy logic* dapat dilihat pada Bab 2.10.

2.2 Mikrokontroler Arduino

Arduino adalah mikrokontroler *single-board* yang bersifat *open-source*, dirancang untuk memudahkan penggunaan purwarupa perangkat elektronik dalam berbagai bidang. Arduino juga menggunakan *Integrated Development Environment (IDE)* berbasis *processing* yang dapat menuliskan program ke komputer lainnya. Jika ada sebuah proyek yang memerlukan beberapa komputer untuk berkomunikasi dengan Arduino, maka *processing* tersebut dapat digunakan sehingga komunikasi antar komputer dapat dilakukan melalui arduino. [3]

Dalam penggunaannya, Arduino memiliki beberapa kelebihan, diantaranya harga yang terjangkau, *multiplatform*, sederhana, dan *open source*. Selain itu, mikrokontroler Arduino bisa dikustomisasi dengan berbagai macam perangkat lain dengan

funksinya masing-masing, misalnya *shield* yang mengakomodasi komunikasi data seperti *Ethernet shield* dan *Bluetooth shield*, atau perangkat sensor seperti misalnya yang digunakan dalam penelitian ini adalah sensor kelembaban tanah dan udara. Mikrokontroler Arduino sendiri memiliki jenis yang beragam seperti Arduino Uno, Arduino Mega 2560, Arduino Leonardo, dan lain-lain. Dalam penelitian ini jenis Arduino yang digunakan adalah jenis Arduino Uno dan Arduino Mega. Alasan penggunaan kedua jenis Arduino tersebut adalah karena masing-masing jenis Arduino memiliki peran dan fungsinya masing-masing, yaitu menjadi *master node* dan *slave node*. Spesifikasi dan penjelasan mengenai kedua jenis Arduino tersebut dapat dilihat lebih lanjut pada Bab 2.3 dan Bab 2.4

2.3 Arduino Uno

Arduino Uno merupakan salah satu jenis turunan dari mikrokontroler Arduino. Arduino Uno menggunakan *chip* mikrokontroler berbasis ATmega 328. Beroperasi dengan tegangan 5V dan SRAM 2Kb. Memiliki 32 KB *flash memory* untuk menyimpan program dan 1 KB EEPROM untuk menyimpan parameter. Kecepatan *clock*-nya 16 MHz, yang setara dengan 300.000 baris dalam sebuah source code C per detiknya. Memiliki 14 pin I/O dan 6 pin *input* analog. [4]



Gambar 2.2. Arduino Uno

Sebagaimana yang ditunjukkan pada Gambar 2.2, Arduino Uno memiliki penghubung USB untuk dapat terhubung dengan komputer dan memungkinkan penggunaan baterai sebagai sumber tenaga pengoperasiannya. Arduino Uno juga memiliki sebuah *power jack* agar Arduino tersebut dapat diberi daya lewat kabel adapter.

Dalam penelitian ini, mikrokontroler Arduino Uno akan digunakan sebagai perangkat yang berperan sebagai *slave node*. Mikrokontroler Arduino inilah yang akan diintegrasikan dengan perangkat-perangkat sensor dan modul *Bluetooth*.

2.4 Arduino Mega 2560

Arduino Mega 2560 adalah salah satu jenis turunan dari mikrokontroler Arduino. Arduino Mega menggunakan *chip* mikrokontroler berbasis ATmega2560. Beroperasi dengan tegangan 5V dengan *range* tegangan *input* 6V-20V. Memiliki 54 buah pin I/O pin digital, 16 pin analog *input*, dan 4 pin UART untuk berkomunikasi. Beroperasi layaknya Arduino Uno sebagaimana yang telah dijelaskan pada Bab 2.3, namun perbedaannya adalah jumlah pin dan kapasitas yang dimiliki oleh mikrokontroler ini. Memiliki flash memory sebesar 256 KB, SRAM 8 KB, dan EEPROM sebesar 4 KB. [5]



Gambar 2.3. Arduino Mega 2560

Sebagaimana yang ditunjukkan pada Gambar 2.3, Arduino Mega 2560 memiliki penghubung USB untuk dapat terhubung dengan komputer dan memungkinkan penggunaan baterai sebagai sumber tenaga pengoperasiannya. Arduino Mega 2560 juga memiliki sebuah *power jack* agar Arduino tersebut dapat diberi daya lewat kabel adapter.

Dalam penelitian ini, mikrokontroler Arduino Mega 2560 akan digunakan sebagai perangkat yang berperan sebagai *master node*. Alasan penggunaan Arduino Mega 2560 yang dipilih sebagai *master node* adalah karena faktor jumlah pin yang ada pada Arduino Mega 2560 tersebut. Bila dibandingkan dengan Arduino Uno yang hanya memiliki 2 pin komunikasi, Arduino Mega 2560 memiliki 4 pin komunikasi sehingga memungkinkan *master node* untuk menjalankan komunikasi dengan *slave node* secara kontinyu, tidak perlu terjadi adanya *interrupt* karena dua komunikasi serial secara sekaligus.

2.5 Bluetooth

Bluetooth adalah sebuah teknologi komunikasi nirkabel yang beroperasi dalam pita frekuensi 2,4 GHz (antara 2.402 GHz sampai dengan 2.480 GHz) dengan menggunakan sebuah *frequency hopping tranceiver* yang mapu menyediakan layanan komunikasi data dan juga suara secara *real-time* antara *host Bluetooth* dengan jarak jangkauan layanan yang terbatas. Dapat digunakan untuk menghubungkan sebuah perangkat komunikasi dengan perangkat komunikasi lainnya, *Bluetooth* umumnya digunakan di telepon genggam, komputer, tablet, dan lain-lain.

Karakteristik dari teknologi *Bluetooth* adalah *Bluetooth* merupakan standar untuk komunikasi nirkabel jarak dekat, murah, dan hemat energy yang menggunakan teknologi radio. Karena itu teknologi *Bluetooth* digunakan dalam sistem karena karakteristiknya yang mendukung tujuan dari penelitian ini, yaitu mendukung sebagai teknologi yang diaplikasikan pada perangkat-perangkat yang ukurannya kecil sebagaimana yang telah disebutkan. Model komunikasi dari *Bluetooth* ini adalah *point-to-*

point, yaitu satu perangkat hanya terhubung dengan satu perangkat pada satu waktu. Namun dalam pengembangannya saat ini, teknologi *Bluetooth* juga dikembangkan tidak hanya model point-to-point tetapi juga sebagai sebuah jaringan. [2]

2.6 GPS/GSM/GPRS *Shield* V3.0

GPS/GSM/GPRS *Shield* V3.0 merupakan *shield* yang diproduksi DFRobot. *Shield* ini bekerja dengan frekuensi Quad-band GSM/GPRS pada 850 MHz, 900 MHz, 1800 MHz, dan 1900 MHz. *Shield* ini juga mendukung teknologi GPS untuk keperluan navigasi satelit sehingga memungkinkan sebuah robot atau sistem mengirim data lokasi melalui jaringan GSM. *Shield* ini dirancang sedemikian rupa sehingga memungkinkan pengguna menjalankan fungsi GPS dan GSM langsung dari komputer dan mikrokontroler Arduino.

Sebagaimana layaknya *shield* lainnya yang diperuntukkan untuk dipasang pada berbagai perangkat, GPS/GSM/GPRS *Shield* V3.0 dibuat dengan pin-pin yang tidak menghalangi masuknya *input* atau *output* pada mikrokontroler. *Shield* langsung dipasangkan begitu saja pada mikrokontroler dengan cara langsung ditimpa dan mencocokkan pin-pin mikrokontroler yang tersedia dengan pin-pin yang ada pada GPS/GSM/GPRS *Shield* V3.0. Tampak rupa dari GPS/GSM/GPRS *Shield* V3.0 dapat dilihat pada Gambar 2.4.



Gambar 2.4. GPS/GSM/GPRS *Shield* V3.0

Shield ini bekerja dengan *command* khusus yang disebut *AT command*. *AT command* akan diprogram dan *shield* akan menjalankan perintah sesuai dengan *command* dalam program. Beberapa contoh penggunaan *AT command* dapat dilihat pada Tabel 2.1 berikut,

Tabel 2.1. Contoh AT Command pada GPS/GPRS/GSM Shield V3.0

No.	AT Command	Keterangan
1	AT+CGREG=?	Mengecek status jaringan
2	AT+HTTTPINIT	Inisiasi mode HTTP
3	AT+HTTTPTERM	Menyudahi mode HTTP
4	AT+HTTTPPARAM	Mengatur URL yang akan dikunjungi

Shield ini menggunakan sistem tertanam SIM908, yang mengkombinasikan teknologi GPS untuk navigasi satelit dan teknologi GPRS dan GSM. [5]

2.7 Modul Bluetooth HC-05

Modul *Bluetooth* HC-05 adalah perangkat keras yang menjalankan modul *Bluetooth* SPP (*Serial Port Protocol*) yang memungkinkan untuk komunikasi antar perangkat lewat koneksi jaringan nirkabel. Modul *Bluetooth* HC-05 memungkinkan sebuah perangkat untuk memiliki perannya masing-masing, yaitu apakah berperan sebagai *master* atau berperan sebagai *slave*. Peran ini dapat diatur sesuai kebutuhan. Berbeda dengan modul *Bluetooth* lain, modul *Bluetooth* HC-05 memungkinkan perangkat untuk dapat bertukar peran, bisa menjadi *master*, dan bisa juga sebagai *slave*. Sementara itu, modul *Bluetooth* yang lain seperti modul *Bluetooth* HC-03, HC-04, dan HC-06 yang hanya mampu berperan salah satu antara *master* atau *slave*.

Modul *Bluetooth* HC-05 memiliki 34 pin yang masing-masing pinnya memiliki peran dan fungsi yang berbeda-beda. Dalam penelitian ini, yang digunakan hanya 5 pin saja, yaitu pin

RXD, TXD, VCC, GND, dan pin 31. Tampak rupa dari modul *Bluetooth* HC-05 dapat dilihat pada Gambar 2.5.



Gambar 2.5. Modul *Bluetooth* HC-05

Pada modul ini juga digunakan *AT command* untuk menjalankan perintah tertentu. Berapa contoh *AT command* yang digunakan dalam modul *Bluetooth* HC-05 dapat dilihat pada tabel 2.2.

Tabel 2.2. Contoh *AT Command* pada HC-05

No.	<i>AT Command</i>	Respon	Parameter	Keterangan
1	AT+RESET	OK	-	Mereset modul
2	AT+RMAAD	OK	-	Menghapus koneksi dari perangkat sebelumnya
3	AT+INIT	OK/FAIL	-	Menginisialisasi SPP
4	AT+LINK=<Parameter>	OK/FAIL	Alamat perangkat	Menghubungkan modul dengan perangkat lain
5	AT+ROLE=<Parameter>	OK	0-Slave; 1-Master	Mengatur <i>role</i> (slave / master)

2.8 Sensor Kelembaban Tanah FC-28

Sensor kelembaban adalah sebuah sensor yang bisa membaca keadaan kelembaban tanah dalam sebuah daerah tertentu. Teknologi yang dipakai sensor ini tergolong sederhana, namun cocok untuk memantau keadaan tanah dalam sebuah lahan. Sensor kelembaban tanah bekerja dengan menggunakan daya hantaran listrik, yakni dengan cara sensor menghantarkan listrik ke tanah untuk kemudian melihat *feedback* dari tanah. Karena tanah yang lembab akan mengandung lebih banyak air maka jika tanah lembab *feedback* dari hantaran listrik tersebut juga akan semakin kuat. Karena itu sensor kelembaban tanah pada umumnya terdiri dari dua perangkat, yaitu perangkat sensor yang langsung menancap ke tanah untuk menghantarkan listrik dan perangkat lain yang bertugas untuk mengkonversi daya listrik *feedback* yang diterima.



Gambar 2.6. Sensor kelembaban tanah FC-28

Sensor kelembaban yang digunakan adalah FC-28. Sebagaimana yang tampak pada Gambar 2.6, sensor kelembaban tanah FC-28 ini dengan dua cabang di sensornya yang nanti akan menancap ke tanah untuk mendapatkan level kelembaban tanah

tersebut. Sensor ini dirancang untuk dapat bekerja dengan tegangan listrik yang digunakan 3.3V hingga 5V.

2.9 Sensor Suhu dan Kelembaban Udara DHT11

Sensor suhu dan kelembaban udara adalah sebuah sensor yang dapat membaca nilai dan kelembaban udara dari suatu tempat.

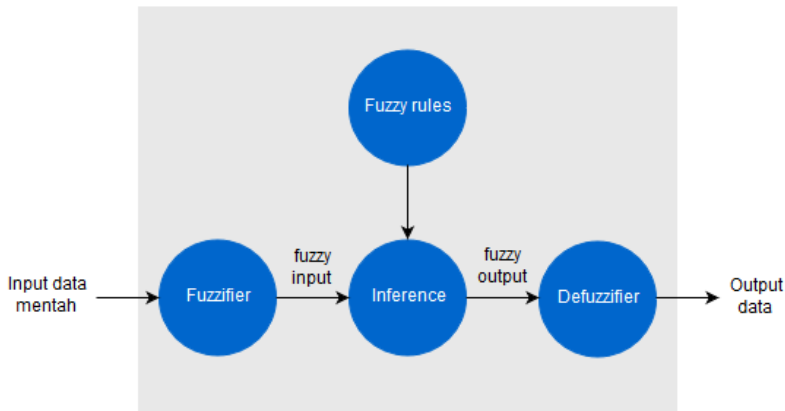


Gambar 2.7. Sensor suhu dan kelembaban udara DHT11

Sensor suhu dan kelembaban udara yang digunakan adalah sensor DHT11. Membaca nilai suatu tempat dalam satuan Celsius dan membaca nilai kelembaban udara dalam satuan persentase RH atau persentasi kelembaban *relative*. Beroperasi dengan menggunakan tegangan 3.3V-5V.

2.10 Metode *Fuzzy logic*

Metode *fuzzy logic* dapat didefinisikan sebagai sebuah metode pemetaan dan pengambilan keputusan dari sejumlah *input* kondisi atau nilai untuk menghasilkan sebuah *output* berupa status atau keputusan. Metode *fuzzy logic* terdiri dari empat bagian utama, yaitu *fuzzifier*, *fuzzy rules*, *inference engine*, dan *defuzzifier*. Cara kerja metode *fuzzy logic* dan keempat bagian tersebut secara ringkas dapat dilihat pada Gambar 2.8.



Gambar 2.8. Cara kerja *fuzzy logic* secara ringkas

Metode *fuzzy logic* diawali dengan mengumpulkan *input data mentah*, yaitu data yang belum melewati proses fuzzifikasi. Fuzzifikasi adalah suatu proses di mana salah satu fungsinya adalah mengubah data mentah tersebut ke dalam *linguistic variables* yang menjadikan data mentah tadi menjadi satu set *fuzzy input*. Selain mengubah data mentah menjadi *fuzzy input*, fuzzifikasi juga mencakup pendefinisian *membership function* yang nanti akan menentukan nilai derajat keanggotaan dari sebuah *fuzzy set*.

Selanjutnya, *fuzzy input* tadi masuk ke dalam tahap *inference*, yaitu pengambilan keputusan terhadap *fuzzy input* dengan berdasarkan aturan yang telah ditetapkan atau *fuzzy rule*. Singkatnya, *Fuzzy rule* adalah kondisi “IF-THEN”, yaitu kondisi jika-maka. Hasil dari proses ini adalah *fuzzy output*.

Terakhir, *fuzzy output* tersebut masuk ke dalam proses *defuzzifikasi*, yaitu proses mengubah *fuzzy output* yang berupa nilai derajat keanggotaan menjadi *output data* yang dapat dipahami sebagai sebuah kesimpulan.

Pada sistem pemantauan kondisi lahan ini, metode *fuzzy logic* digunakan untuk mengimplementasi fungsi adaptif sistem terhadap lingkungan sistem.

2.11 Kelembaban Lahan

Kelembaban lahan adalah objek yang diamati dalam sistem ini. Kondisi kelembaban lahan secara umum diukur oleh banyak faktor, namun dalam sistem ini yang factor yang diukur dan dijadikan patokan kondisi lahan adalah nilai kelembaban tanah dan nilai kelembaban udara.

Kelembaban tanah didefinisikan sebagai nilai menyatakan jumlah air yang tersimpan di antara pori-pori tanah. kelembaban tanah sangat dinamis, hal ini disebabkan oleh penguapan melalui permukaan tanah, transpirasi atau pernapasan tumbuhan, dan perkolasi atau meresapnya air ke lapisan tanah yang lebih dalam [6].

Nilai kelembaban tanah yang dibaca oleh sensor kelembaban tanah memiliki *range* nilai 0-1050. Dari rentang nilai tersebut, kondisi kelembaban tanah dapat dikategorikan menjadi tiga [7], yaitu kering, baik, dan lembab. *Range* dari masing-masing kategori dapat dilihat pada Tabel 2.3.

Tabel 2.3. Kategori tanah berdasarkan hasil pembacaan sensor

Kategori	Nilai
Kering	0-300
Baik	300-700
Lembab	700-1050

Beberapa hal yang mengubah nilai kelembaban tanah diantaranya besarnya penguapan sehingga membuat air yang ada pada permukaan tanah menguap. Karena itu cara pengendalian yang dapat dilakukan untuk menjaga kondisi kelembaban tanah adalah dengan pengairan untuk memastikan tanah ada pada kondisi kelembaban terbaiknya.

Sementara itu, kelembaban udara yang dibaca pada sensor adalah kelembaban relatif, di mana kelembaban relatif adalah istilah yang digunakan untuk menggambarkan jumlah uap air yang terkandung di dalam campuran air-udara dalam fase gas.

Nilai kelembaban relatif yang dibaca sensor dalam bentuk persen, dari 0-100 persen. Nilai kelembaban relatif udara yang baik tergantung kebutuhannya [8]. Misal kelembaban udara yang baik untuk sebuah ruang penyimpanan buah-buahan berbeda dengan nilai kelembaban udara yang baik untuk lahan pertanian. Untuk lahan pertanian sendiri, kondisi kelembaban udara yang baik terletak pada nilai 40%-60% [9]. Untuk lebih jelasnya dapat dilihat pada Tabel 2.4

Tabel 2.4. Kategori kelembaban udara berdasarkan nilai kelembaban relatif

Kategori	Nilai
Kering	0-40%
Baik	40%-60%
Lembab	60-100%

Beberapa hal yang dapat mengubah nilai kelembaban relatif adalah ketersediaan air pada permukaan dan penguapan. Pada kondisi ketersediaan air permukaan yang tinggi, maka kelembaban relatif udara akan ikut naik pula mengingat akan banyak air yang menguap ke udara. Karena itu cara pengendalian kelembaban udara yang banyak digunakan adalah penggunaan ventilasi untuk mengatur keluar masuknya udara sehingga sirkulasi udara baik dan penguapan yang terjadi dapat dikendalikan.

Kedua faktor tersebut, yaitu kelembaban tanah dan kelembaban udara dapat saling terkait satu sama lain. Karena perubahan salah satunya bisa jadi mengubah yang lainnya. Misal pada kelembaban tanah yang tinggi atau lembab, kelembaban relatif udara pada umumnya akan tinggi pula mengingat adanya penguapan, sehingga penanganan kondisi kelembaban tanah akan berimbas pada kondisi kelembaban udara. Contoh lain ketika kondisi kelembaban tanah ada pada kondisi kering sehingga butuh pengairan, namun kondisi kelembaban udara ada pada kondisi baik. Yang harus dalam diperhatikan adalah untuk mengatasi kondisi kelembaban tanah yang kering yang membutuhkan air

tersebut, dibutuhkan pengairan. Akan tetapi pengairan yang dilakukan pun harus memperhatikan kondisi kelembaban *relative* udara. Jangan sampai pengairan untuk mengatasi kondisi kelembaban tanah yang kering menjadikan kondisi kelembaban *relative* udara yang awalnya baik menjadi terlalu lembab, Kondisi yang saling terkait inilah yang akan menjadi pertimbangan dalam penentuan status kondisi lahan pada sistem.

(Halaman ini sengaja dikosongkan)

BAB III

PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai perancangan sistem, baik perangkat keras maupun lunak. Perancangan merupakan bagian penting dari pembuatan perangkat lunak yang berupa perencanaan-perencanaan secara teknis aplikasi yang dibuat. Sehingga bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir ini. Berawal dari deskripsi umum aplikasi hingga perancangan proses, alur dan implementasinya.

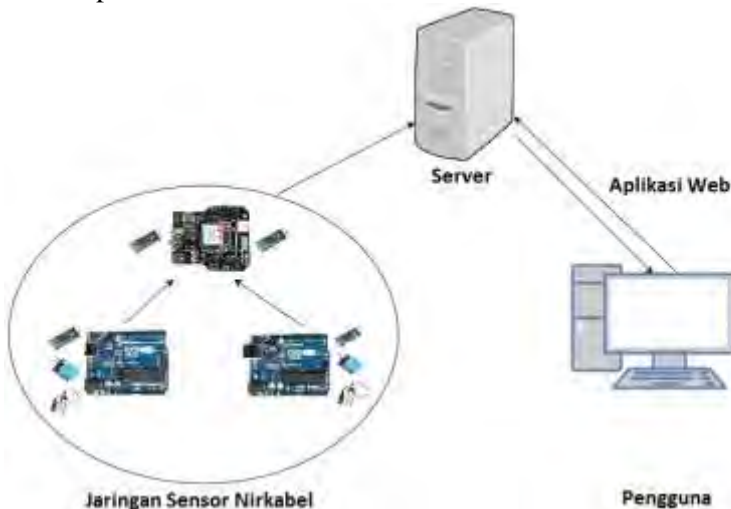
3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun suatu sistem pemantauan kondisi kelembaban lahan yang hemat energi, ekonomis, dan dapat bekerja secara adaptif. Dirancang untuk melakukan pemantauan dengan menggunakan indikator berupa data kelembaban tanah dan kelembaban udara yang kemudian menjadi *trigger* untuk melakukan pengiriman data.

Dibutuhkan dua jenis sensor untuk mendapatkan data yang menjadi *trigger* dalam pengembangan sistem ini, yaitu sensor kelembaban tanah, dan sensor suhu dan kelembaban udara, yang diintegrasikan dengan mikrokontroler Arduino. Untuk menentukan kapan data dari sensor akan dikirimkan, sistem ini melakukan agregasi data yang dilakukan secara adaptif dengan *output* kondisi lahan yang dipantau yang dihasilkan dari metode *fuzzy logic* dengan *input* nilai kelembaban tanah, , dan kelembaban udara. Pengiriman data hanya akan dilakukan bila terdapat perubahan yang signifikan pada kondisi lahan. Data akan dikirimkan ke *server* untuk kemudian dapat diakses oleh pengguna lewat koneksi internet. Aplikasi web berperan membaca hasil akhir dari pengolahan data pada sistem tersebut.

3.2 Arsitektur Umum Sistem

Rancangan arsitektur umum dari sistem yang dibuat dapat dilihat pada Gambar 3.1



Gambar 3.1. Rancangan arsitektur umum

Berdasarkan perancangan arsitektur sistem pada Gambar 3.1, pengambilan data kondisi lahan dilakukan oleh jaringan sensor nirkabel berbasis *Bluetooth*. Jaringan sensor nirkabel terdiri dari dua buah *slave node* yang masing-masing terintegrasi dengan sensor kelembaban tanah dan sensor suhu dan kelembaban udara, dan terhubung dengan sebuah *master slave* dengan menggunakan koneksi *Bluetooth* dari *Bluetooth module*. Selanjutnya, data yang diterima oleh *master slave* akan dikirim menuju *server* dengan menggunakan *GPS/GSM/GPRS Shield* untuk kemudian dapat diakses oleh pengguna dengan menggunakan koneksi internet.

Sifat adaptif pada sistem ini terdapat pada pengiriman data, yaitu pengiriman data dari masing-masing *slave node* menuju *master node*, lalu pengiriman data dari *master node* menuju *server*.

Sifat adaptif pada pengiriman data dari *slave node* menuju *master node* ditentukan dari metode *fuzzy logic* yang menghasilkan

output kondisi lahan, dari *input* data kelembaban tanah dan kelembaban udara.

3.3 Perancangan Perangkat Keras

Perangkat keras digunakan dalam membangun jaringan sensor nirkabel berbasis *Bluetooth*. Dalam satu jaringan sensor nirkabel tersebut terdapat *node-node* yang memiliki peran masing-masing, yaitu *master node* dan *slave node*.

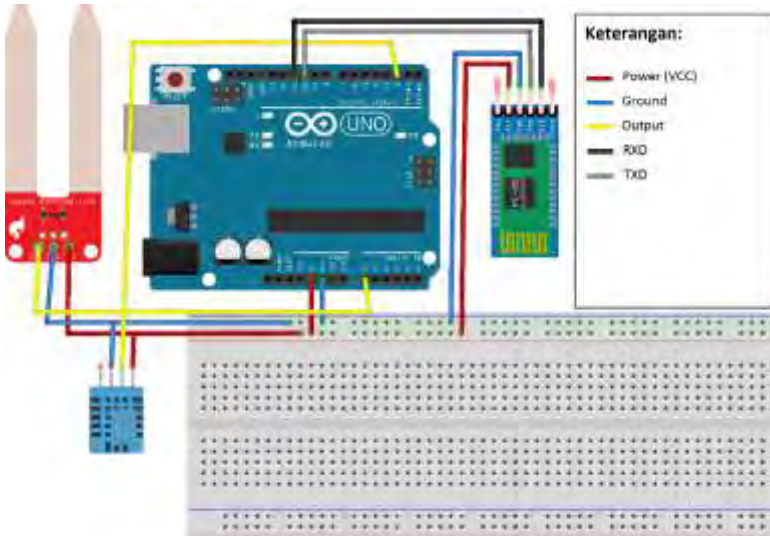


Gambar 3.2. Perancangan perangkat keras

3.3.1 Perancangan pada *Slave Node*

Secara umum, dapat didefinisikan bahwa *slave node* adalah *node-node* yang tersebar di dalam sebuah jaringan sensor nirkabel sebagaimana yang dijelaskan pada Bab 2.1. Sehingga peran dan fungsi *slave node* adalah *node* yang bertugas untuk mengumpulkan data sensor untuk kemudian dikirim menuju *gateway*, yang dalam penelitian ini disebut *master node*. Penyebutan *slave node* mengacu pada peran modul *Bluetooth* yang diinisialisasi, yaitu sebagai *slave*. *Slave node* dibangun dari sebuah mikrokontroler Arduino Uno yang diintegrasikan dengan sensor kelembaban tanah FC-28 dan sensor DHT11, serta modul

Bluetooth HC-05. Perancangan pada *slave node* dapat dilihat pada Gambar 3.3.



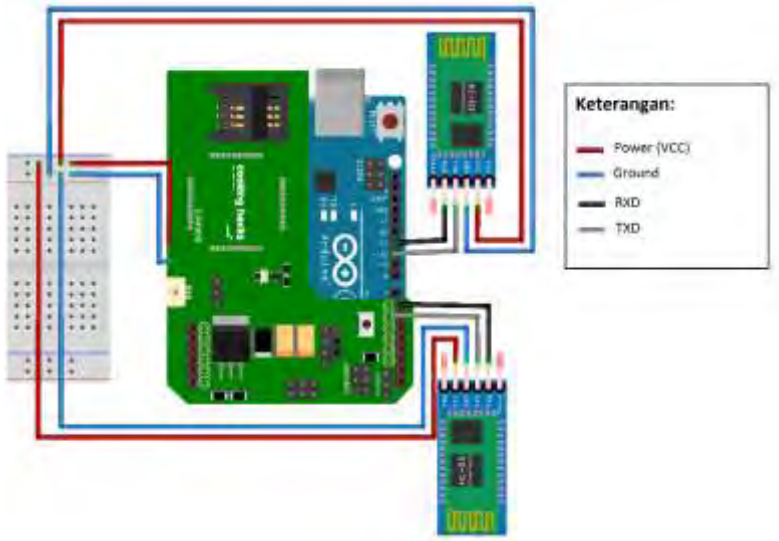
Gambar 3.3. Perancangan pada *slave node*

Mikrokontroler Arduino bertugas menjalankan program yang telah diinisialisasi ke dalam Arduino. Sensor FC-28 bertugas untuk mendapatkan data nilai kelembaban tanah. Sensor DHT11 bertugas untuk mendapatkan data nilai dan kelembaban udara. Modul *Bluetooth* HC-05 bertugas untuk mengirim data dari *slave node* menuju *master node*.

3.3.2 Perancangan pada *Master Node*

Master node adalah *node* yang berperan sebagai *gateway* pada jaringan sensor nirkabel. *Master node* dibangun dari sebuah mikrokontroler Arduino Mega 2560 yang diintegrasikan dengan dua buah modul *Bluetooth* HC-05 dan sebuah GPS/GSM/GPRS

Shield V3.0. Perancangan pada *master node* dapat dilihat pada Gambar 3.4.



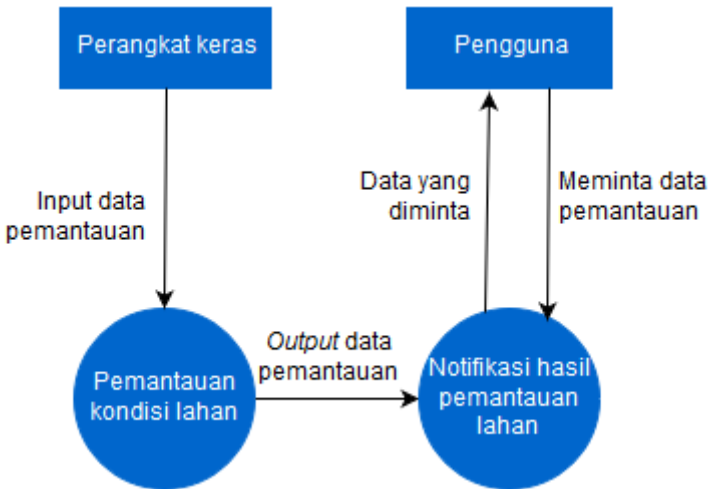
Gambar 3.4. Perancangan pada *master node*

Modul *Bluetooth* HC-05 bertugas untuk berkomunikasi dengan *slave node*. *GPS/GSM/GPRS Shield* bertugas untuk mengirim data menuju *server*.

3.4 Perancangan Diagram Alir Data *Level 0*

Diagram alir data *level 0* menggambarkan fungsionalitas sistem dengan aktor yang terlibat dalam sistem tersebut. Aktor yang terlibat dalam sistem pemantauan kondisi lahan ini adalah pengguna. Sistem diawali dengan pengumpulan data pemantauan yang didapat dari jaringan sensor nirkabel berbasis *Bluetooth*. Data yang dikumpulkan adalah data nilai kelembaban tanah yang didapat dari sensor FC-28, dan data suhu serta kelembaban udara

dari sensor DHT11. Kemudian data yang dikumpulkan tersebut diintegrasikan dengan menggunakan metode *fuzzy logic* yang menghasilkan *output* data hasil pemantauan. Diagram alir data level 0 pada sistem pemantauan kondisi lahan ini dapat dilihat pada Gambar 3.5.



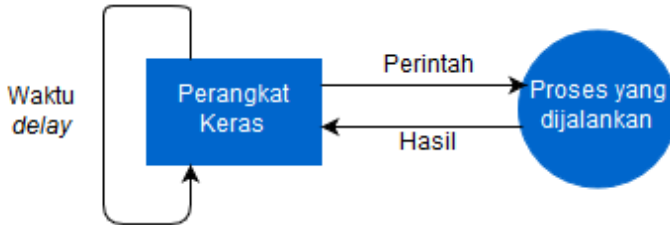
Gambar 3.5. Diagram alir data pada level 0

Hasil akhirnya sebagaimana yang terlihat pada Gambar 3.5, *Output* data hasil pemantauan selanjutnya dikirimkan ke *server* dan disimpan ke dalam *database*. Data hasil pemantauan kondisi lahan dapat diakses oleh pengguna lewat aplikasi web. Sistem pemantauan kondisi lahan ini berjalan dengan otomatis.

3.5 Perancangan Sifat Adaptif Sistem

Sistem pemantauan kondisi lahan ini berjalan dengan otomatis, artinya segala proses yang terjadi berjalan dengan sendirinya di semua tahapan mulai dari pengumpulan, pembacaan, pengolahan, hingga pengiriman data. Kerja sistem yang otomatis bukan berarti sistem beroperasi tanpa adanya aturan-aturan atau kondisi tertentu yang membuat tahapan-tahapan dalam sistem

tersebut berjalan. Pasti ada perintah yang diterima sistem sebagai prakondisi dari tahap yang dijalankan. Diagram alir berjalannya sistem secara umum dapat dilihat pada Gambar 3.6.

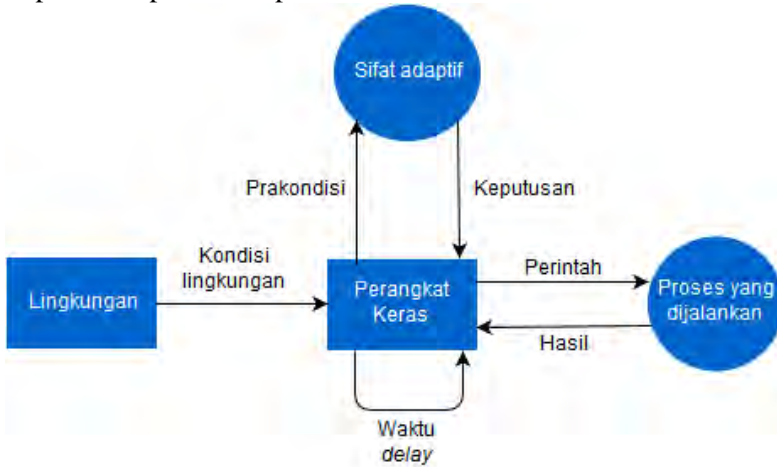


Gambar 3.6. Diagram alir berjalannya sistem secara umum

Sistem berjalan dengan satuan waktu tertentu yang disebut waktu *delay*. Jika hanya bergantung pada waktu *delay*, sistem akan berjalan dengan sendirinya namun tidak menyesuaikan diri dengan kondisi lingkungan sistem. Artinya sistem terus beroperasi bagaimanapun kondisi lingkungan sistem, mengirim perintah dari perangkat dan menjalankan proses yang diinginkan untuk kemudian perangkat menerima hasil dari proses tersebut, yang berjalan sesuai dengan waktu *delay* yang telah ditentukan.

Cara kerja sistem yang sedemikian rupa memiliki kekurangan diantaranya jalannya sistem yang tidak menyesuaikan dengan kondisi lingkungan akan membuat sistem menjalankan fungsi-fungsi yang tidak perlu. Contohnya pada sistem pemantauan kondisi lahan ini adalah jika sistem terus melakukan transmisi pengiriman data menuju *server* dalam suatu rentang waktu *delay* tertentu padahal kondisi lahan yang dipantau tersebut tidak menunjukkan perubahan yang signifikan. Artinya *server* terus menerima dan menyimpan data yang sama secara berulang kali. Hal ini tentu tidak efektif dan efisien mengingat sumber daya yang dipakai terbatas. Karena itu sistem membutuhkan penjadwalan agar sistem dapat bekerja secara efektif dan efisien.

Penjadwalan ini menjadikan sistem bekerja dengan sifat adaptif. Gambaran penjadwalan sistem yang bekerja dengan sifat adaptif ini dapat dilihat pada Gambar 3.7.



Gambar 3.7. Gambaran sifat adaptif sistem

Perangkat keras menerima data kondisi dari lingkungan implementasi sistem, kemudian data tersebut dijadikan prakondisi untuk kemudian dicocokkan dengan sifat adaptif yang dimiliki sistem. Melihat pada aturan sifat adaptif tersebutlah keputusan diambil apakah perangkat mengirimkan perintah untuk menjalankan proses selanjutnya dalam sistem atau tidak. Dengan aturan seperti ini, perangkat tidak hanya bekerja sesuai dengan waktu *delay* yang ditentukan, tetapi juga kondisi lingkungan sistem.

Dalam sistem pemantauan kondisi lahan ini, sifat adaptif sistem ditentukan dengan metode *fuzzy logic*. Sifat adaptif dengan metode *fuzzy logic* ini akan diimplementasikan pada proses yang terjadi di perangkat keras, yaitu pada *master node* dan *slave node*. Mengenai langkah-langkah dalam implementasinya dapat dilihat pada Bab 3.6.5 dan Bab 3.6.6.

3.6 Perancangan *Fuzzy Logic*

Fuzzy logic memegang peran yang amat penting dalam sistem ini, mengingat *fuzzy logic* berperan sebagai aturan penjadwalan dalam sistem yang membuat sistem bekerja secara adaptif. *Fuzzy logic* inilah yang akan memutuskan kapan pengiriman data akan dilakukan setelah melihat perubahan kondisi pada lahan yang dipantau.

Fuzzy logic diimplementasikan pada dua tempat, yaitu pada *slave node* dan juga pada *master node*.

3.6.1 Perancangan *Fuzzy Logic* pada *Slave Node*

Fuzzy logic sebagaimana yang telah dijelaskan dalam Bab 2.9, bekerja dengan cara mengolah sejumlah data masukan untuk selanjutnya melalui sejumlah proses dan kemudian menghasilkan sebuah data keluaran.

Slave node bekerja dengan mengumpulkan data nilai kelembaban tanah dan kelembaban udara yang didapat dari dua sensor yang diintegrasikan pada *slave node*. Dua nilai sensor inilah yang menjadi *input* untuk *fuzzy logic* pada *slave node*.

Input yang pertama yaitu data nilai kelembaban tanah. *Range* nilai kelembaban yang terbaca oleh sensor adalah 0-1023. Untuk dapat digunakan sebagai *fuzzy input*, nilai tersebut harus dijadikan sebagai variabel linguistik. Nilai kelembaban tanah yang telah dijadikan variabel linguistik dapat dilihat pada Tabel 3.1.

Tabel 3.1. Variabel Linguistik Kelembaban Tanah

Variabel	Nilai
Kering	700-1000
Baik	300-750
Lembab	0-400

Sementara itu, nilai kelembaban udara yang didapat dari sensor DHT11 juga dijadikan variabel linguistik. Nilai kelembaban

udara yang telah dijadikan variabel linguistic dapat dilihat pada Tabel 3.2.

Tabel 3.2. Variabel linguistik kelembaban udara

Variabel	Nilai
Kering	0-45%
Baik	40%-70%
Lembab	65%-100%

Kedua variabel linguistik tersebut akan digunakan sebagai *fuzzy input*, yang akan menghasilkan *fuzzy output* sesuai dengan nilai derajat keanggotaannya. Gambaran mengenai hubungan antara *fuzzy input* dengan *fuzzy output* dapat dilihat pada matriks *fuzzy* yang ditunjukkan pada Tabel 3.3.

Tabel 3.3. Matriks Fuzzy pada slave node

Udara\Tanah	Kering	Baik	Lembab
Kering	Sangat Kering	Butuh Pengairan	x
Baik	Butuh Pengairan	Baik	Butuh Ventilasi
Lembab	x	Butuh Ventilasi	Sangat Lembab

Dua *fuzzy input*, yaitu kelembaban tanah dan kelembaban menghasilkan lima kondisi yang juga digunakan sebagai *fuzzy output*. Perubahan dari kelima kondisi inilah yang akan dijadikan parameter untuk pengiriman data, apakah ada perubahan status kondisi atau tidak. Jika terjadi perubahan status maka status terbaru akan dikirim kepada *master node* untuk kemudian diproses lebih lanjut. Lima status kondisi yang dihasilkan, yaitu sangat kering, butuh pengairan, baik, butuh ventilasi, dan sangat lembab merupakan status kondisi kelembaban sebuah ruang indoor yang umum dikenal. [10]

Fuzzy input dan *fuzzy output* yang sudah ditentukan sebelumnya kemudian diformulasikan menjadi *fuzzy rule*. Secara sederhana, *fuzzy rule* yang akan diimplementasikan pada *slave node* dapat pada Tabel 3.4

Tabel 3.4. *Fuzzy rule* pada *slave node*

Antecedent	Consequence
Jika tanah kering DAN udara kering	Sangat kering
Jika tanah kering DAN udara baik ATAU tanah baik DAN udara kering	Butuh pengairan
Jika tanah baik DAN udara baik	Baik
Jika tanah baik DAN udara lembab ATAU tanah lembab DAN udara baik	Butuh ventilasi
Jika tanah lembab DAN udara lembab	Sangat lembab

Fuzzy rule yang ada pada sistem mengikuti kondisi yang ada di lapangan, hal ini dapat tergambar pada matriks *fuzzy* pada Tabel 3.3 dan *fuzzy* pada Tabel 3.4. Pada matriks *Fuzzy* dapat dilihat ada beberapa kondisi di mana kedua *input* tidak menghasilkan *output* apa-apa. Hal ini terjadi karena kondisi demikian hampir dikatakan mustahil atau pun jika terjadi maka kondisi tersebut adalah kondisi anomali. Sebagaimana yang dijelaskan pada Bab 2.11 kondisi kelembaban tanah dan udara dapat memengaruhi dan saling terkait satu sama lain. Kondisi di mana tanah kering DAN udara lembab ATAU tanah lembab DAN udara kering adalah kondisi anomali. Kondisi di mana udara lembab namun tanah kering atau tanah lembab namun udara kering adalah anomali karena berkaitan dengan penguapan tanah dan konsentrasi kadar air di udara.

3.6.2 Perancangan *Fuzzy Logic* pada *Master Node*

Master node menerima data dari kedua *slave node*. *Master node* kemudian menentukan apakah akan mengirimkan status

lahan kepada *server* atau tidak, tergantung *input* yang masuk dari *slave node*. *Slave node* mengirim data berupa status, yang kemudian langsung dijadikan sebagai *fuzzy input*. Kedua *input* tersebut kemudian akan menghasilkan sebuah *output*. Hubungan antara *input* dan *output* pada *master node* dapat dilihat pada Tabel 3.4.

Tabel 3.5. Matriks Fuzzy pada master node

Node 1\Node2	Sangat Kering	Butuh Pengairan	Baik	Butuh Ventilasi	Sangat Lembab
Sangat Kering	Sangat Kering	Sangat Kering	Butuh Pengairan	x	x
Butuh Pengairan	Sangat Kering	Butuh Pengairan	Butuh Pengairan	x	x
Baik	Butuh Pengairan	Butuh Pengairan	Baik	Butuh Ventilasi	Butuh Ventilasi
Butuh Ventilasi	x	x	Butuh Ventilasi	Butuh Ventilasi	Sangat Lembab
Sangat Lembab	x	x	Butuh Ventilasi	Sangat Lembab	Sangat Lembab

Hasil *input* tersebut menghasilkan lima status *output*, yang sama seperti sebelumnya. Namun bedanya *master node* mengkombinasikan data masukan dari dua *node* yang diterima. Jika status hasil *output* sama dengan status sebelumnya, maka data tidak akan terkirim kepada *server*. *Master* hanya mengirim data jika terjadi perubahan status, sehingga yang tersimpan pada *server* adalah kondisi terakhir perubahan lahan.

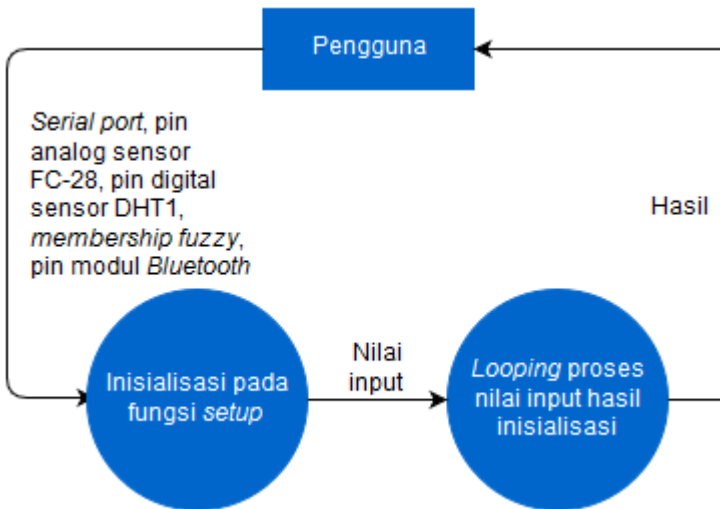
3.7 Diagram Alir Aplikasi Sistem

Diagram alir aplikasi sistem berguna untuk menggambarkan alur pada setiap proses yang terjadi di dalam sistem dan memudahkan pemahaman secara garis besar proses yang ada pada sistem. Diagram alir aplikasi sistem terdiri dari diagram alir

inisialisasi arduino pada *slave node*, diagram alir inisialisasi Arduino pada *master node*, diagram alir mendapatkan nilai kelembaban tanah, diagram alir mendapatkan nilai suhu dan kelembaban udara, diagram alir pengiriman data menuju *master node* secara adaptif, dan diagram alir pengiriman data menuju *server* secara adaptif.

3.7.1 Diagram Alir Inisialisasi Arduino pada *Slave Node*

Saat pertama kali Arduino dinyalakan, Arduino akan melakukan inisialisasi pada fungsi *setup*. Apa yang diinisialisasi pada fungsi *setup* ini hanya dilakukan sekali.

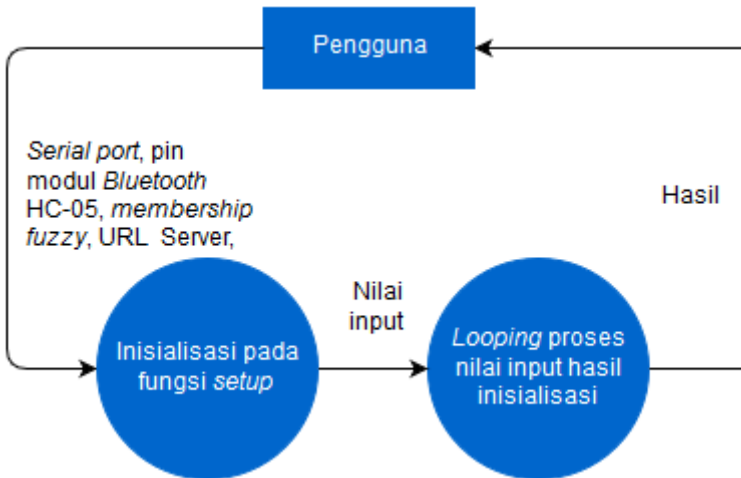


Gambar 3.8. Diagram alir inisialisasi Arduino pada *slave node*

Seperti yang ditunjukkan pada Gambar 3.8, inisialisasi Arduino pada *slave node* meliputi inisialisasi *serial port*, pin analog sensor FC-28, pin digital sensor DHT11, metode *fuzzy logic*, dan pin modul *Bluetooth* HC-05. Inisialisasi *slave node* dilakukan dengan meng-upload source code yang dibuat pada IDE Arduino menuju perangkat mikrokontroler Arduino Uno dengan menggunakan kabel USB yang terhubung dengan komputer.

3.7.2 Diagram Alir Inisialisasi Arduino pada *Master Node*

Inisialisasi Arduino pada *master node* hampir serupa dengan inisialisasi yang dilakukan pada *slave node*, yang menjadi perbedaannya adalah nilai yang diinisialisasi. Diagram alir inisialisasi Arduino pada *master node* dapat dilihat pada Gambar 3.9.



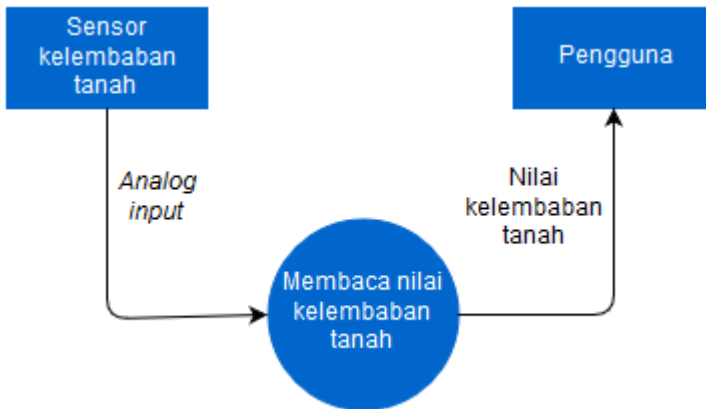
Gambar 3.9. Diagram alir inisialisasi Arduino pada *master node*

Inisialisasi Arduino pada *master node* meliputi inisialisasi serial port, pin modul *Bluetooth* HC-05, *fuzzy logic*, dan URL untuk mengirimkan data menuju *server*. Inisialisasi untuk *master node* dilakukan dengan cara meng-upload source code yang telah dibuat pada IDE Arduino menuju perangkat mikrokontroler Arduino Mega 2560 dengan menggunakan kabel USB yang terhubung dengan komputer. Akan tetapi, tidak seluruhnya proses inisialisasi dilakukan lewat satu perintah upload saja. Untuk inisialisasi modul *Bluetooth*, dilakukan secara terpisah dengan menggunakan *AT Command*. Inisialisasi modul *Bluetooth* dengan menggunakan *AT Command* ini pun tidak sekadar dengan menggunakan kabel USB, tetapi ada sebuah pin pada modul *Bluetooth* yang harus diberi

tegangan untuk dapat masuk ke dalam AT mode agar bias menginputkan AT Command. Inisialisasi pada modul *Bluetooth* lewat AT Command dilakukan untuk menentukan peran dari modul *Bluetooth* itu, apakah akan menjadi *slave* atau menjadi *master*.

3.7.3 Diagram Alir Mendapatkan Nilai Kelembaban Tanah

Nilai kelembaban tanah didapat dari perangkat sensor kelembaban tanah yang bekerja dengan menggunakan hantaran listrik untuk kemudian menerima feedback dari tanah, yang dikonversi menjadi nilai kelembaban tanah yang dapat dibaca menjadi nilai kelembaban tanah mentah. Ketika nilai sensor diperoleh, nilai tersebut dibaca dan kemudian disajikan kepada pengguna. Diagram alir mendapatkan nilai kelembaban tanah dapat dilihat pada Gambar 3.10.

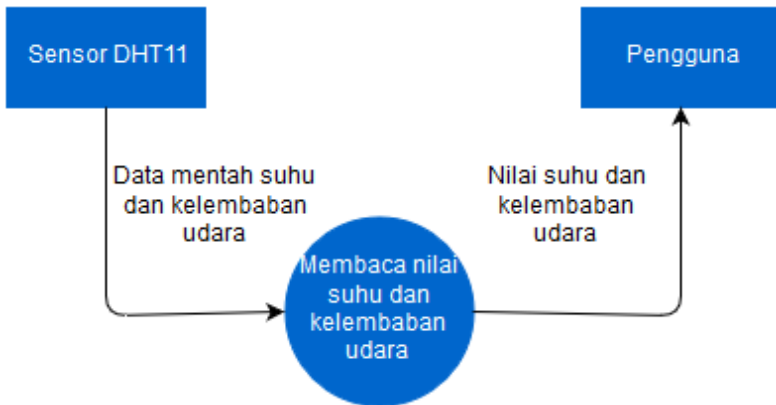


Gambar 3.10. Diagram alir mendapatkan nilai kelembaban tanah

3.7.4 Diagram Alir Mendapatkan Nilai Suhu dan Kelembaban Udara

Nilai suhu dan kelembaban udara didapat dari *digital pin* perangkat sensor DHT11. Sensor DHT11 melakukan pembacaan suhu dan kelembaban udara dengan cara bergantian. Di mana suhu dan udara dipecah ke dalam potongan data 8 bit yang kemudian

dibaca untuk mendapatkan hasil nilai suhu dan kelembaban udara. Namun dalam penelitian ini, data nilai sensor yang digunakan adalah data nilai kelembaban udara, yang disajikan dengan satuan presentase RH (*Relative Humidity*/Kelembaban Relatif). Diagram alir mendapatkan nilai suhu dan kelembaban udara dapat dilihat pada Gambar 3.11.

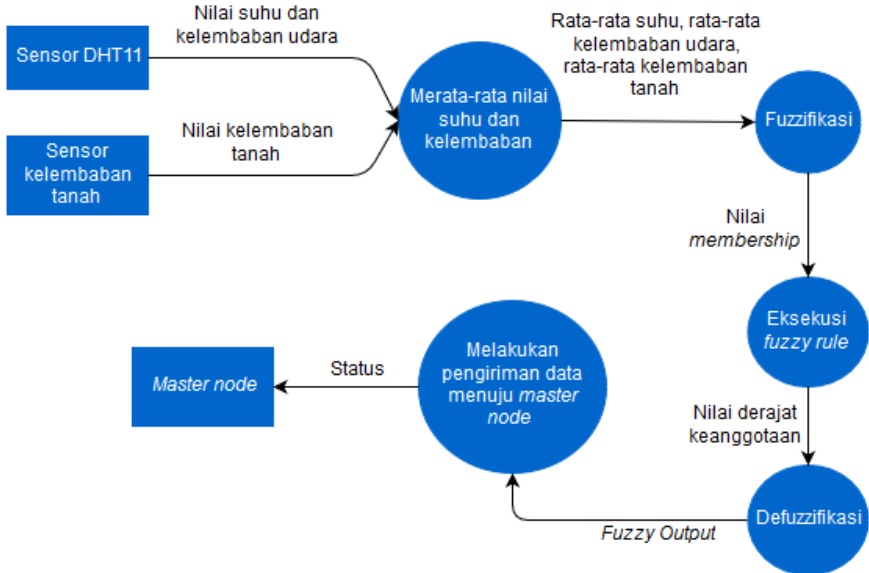


Gambar 3.11. Diagram alir mendapatkan nilai suhu dan kelembaban udara

3.7.5 Diagram Alir Pengiriman Data Menuju *Master Node* Secara Adaptif

Sesuai perannya di dalam suatu jaringan sensor nirkabel, *-node-node* yang tersebar dalam sebuah jaringan yang bertugas untuk mengumpulkan data selanjutnya akan dikirim menuju *gateway*. Begitu pun dengan dalam penelitian ini, *slave node* yang berada pada jaringan bertugas untuk mengirimkan datanya kepada *master node*. Setelah nilai suhu, kelembaban udara, dan kelembaban tanah diterima oleh *slave node*, maka selanjutnya yang dilakukan adalah *slave node* mengirimkan data menuju *master node* dengan koneksi *Bluetooth*. Data yang dikirim oleh *slave node* adalah data yang berbentuk *char* yang merupakan representasi dari

status kelembaban lahan yang dipantau. Pengiriman ini pun tidak dilakukan begitu saja, pengiriman yang dilakukan oleh *slave node* menuju *master node* dilakukan secara adaptif. Diagram alir pengiriman data menuju *master node* secara adaptif dapat dilihat pada Gambar 3.12.

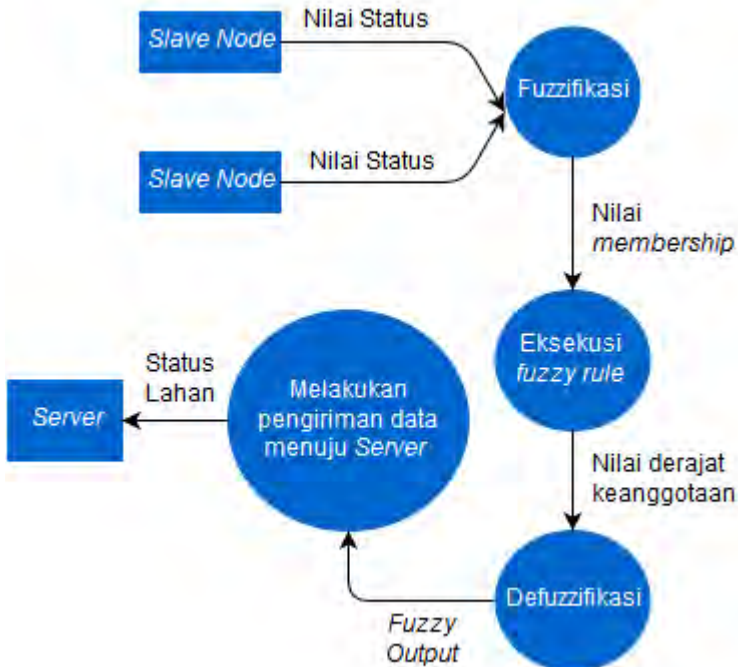


Gambar 3.12. Diagram alir pengiriman data menuju *master node* secara adaptif

Pengiriman data dilakukan secara adaptif memiliki definisi sebagai pengiriman data yang dilakukan sesuai dengan kondisi yang terjadi. Untuk mengetahui kapan pengiriman dilakukan maka digunakan metode *fuzzy logic* untuk menentukan status kondisi kelembaban lahan. Pengiriman hanya akan dilakukan bila terdapat perbedaan yang signifikan pada kondisi lahan dari kondisi yang sebelumnya.

3.7.6 Diagram Alir Pengiriman Data Menuju *Server* Secara Adaptif

Setelah *master node* menerima data status dari *slave node*, *master node* harus kembali mengagregasi data untuk kemudian dikirim menuju *server*. Pengiriman menuju *server* pun bersifat adaptif, yaitu menyesuaikan dengan kondisi lahan. Pengiriman terjadi jika ada perubahan yang signifikan terhadap kondisi lahan yang dipantau. Diagram alir pengiriman data menuju *server* secara adaptif dapat dilihat pada Gambar 3.13.

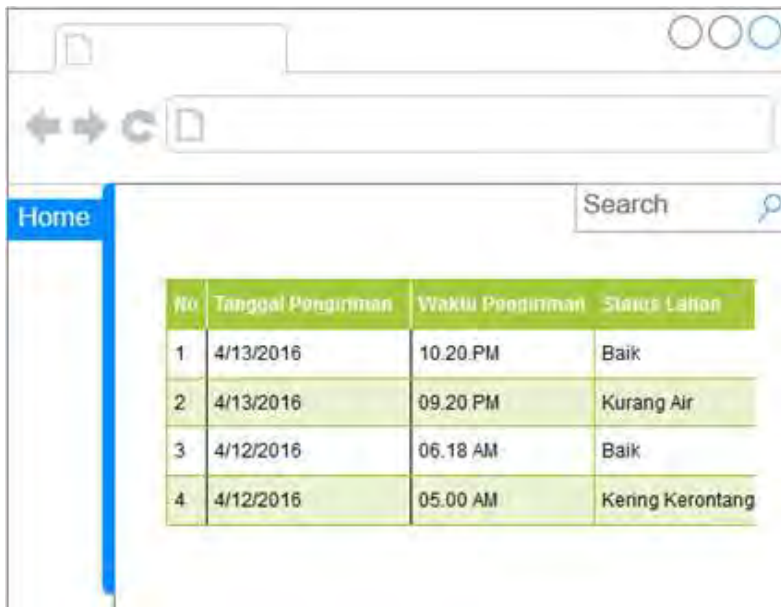


Gambar 3.13. Diagram alir pengiriman data menuju *server* secara adaptif

Dalam menentukan kapan data akan dikirim, *master node* juga menggunakan metode *fuzzy logic* untuk mengetahui apakah ada perubahan yang signifikan terhadap lahan yang dipantau atau tidak berdasarkan *input* yang diterima oleh *server*.

3.8 Rancangan Antarmuka Pengguna

Pengguna dapat melihat aplikasi sistem pemantauan kondisi lahan ini lewat sebuah web. Aplikasi ini memperbaharui data secara otomatis sesuai dengan data yang diterima oleh *server* dari *master node*. Rancangan antarmuka pengguna dapat dilihat pada Gambar 3.14.



No	Tanggal Pengiriman	Waktu Pengiriman	Status Lahan
1	4/13/2016	10.20 PM	Baik
2	4/13/2016	09.20 PM	Kurang Air
3	4/12/2016	06.18 AM	Baik
4	4/12/2016	05.00 AM	Kering Kerontang

Gambar 3.14. Rancangan antarmuka pengguna

Seperti yang tampak pada Gambar 3.14, aplikasi menampilkan tabel yang menunjukkan tanggal pengiriman, waktu pengiriman, dan status lahan. Tanggal pengiriman adalah tanggal *master node* mengirimkan data ke *server*. Waktu pengiriman

adalah waktu *master node* mengirimkan data ke *server*. Status lahan menunjukkan status lahan yang terpantau dan dikirim oleh *master node*.

Selain itu terdapat pula form *Search* untuk mencari data yang ada pada sistem, yakni pengguna memasukkan *keyword* data yang dicari.

BAB IV IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi sistem pada perangkat keras dan perangkat lunak. Implementasi dilakukan berdasarkan perancangan sistem sebagaimana telah dibahas pada Bab3.

4.1 Lingkungan Implementasi

Pengimplementasian sistem pemantauan kondisi lahan ini menggunakan berbagai perangkat, yaitu perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengimplementasian sistem ini adalah komputer dan mikrokontroler Arduino. Spesifikasi lingkungan implementasi perangkat keras yang digunakan dapat dilihat pada Tabel 4.1.

Tabel 4.1. Spesifikasi lingkungan implementasi perangkat keras

Perangkat Keras	Spesifikasi	
Laptop Toshiba Satellite S40-A	Sistem operasi	Windows 8.1 32-bit
	RAM	2GB
	Processor	Intel®Core™i3-3217 1.8GHz
Arduino Uno R3	Microcontroller	ATmega328
	Tegangan	5V
	Flash Memory	32 KB
	SRAM	2 KB

4.1.2 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan sistem ini dari tahap awal hingga tahap uji coba dapat dilihat pada Tabel 4.2.

Tabel 4.2. Spesifikasi lingkungan implementasi perangkat lunak

Sistem Operasi	Windows 8.1 32-bit
Arduino IDE	Arduino 1.6.7
Database Server	MySQL 5.5.50
Web Browser	Mozilla Firefox 46.0.1

4.2 Implementasi Perangkat Keras

Implementasi perangkat keras pada sistem pemantauan kondisi kelembaban lahan ini membutuhkan berbagai perangkat keras untuk dapat bekerja dengan baik dan optimal. Perangkat keras yang dibutuhkan dalam implementasi sistem ini dapat dilihat pada Tabel 4.3.

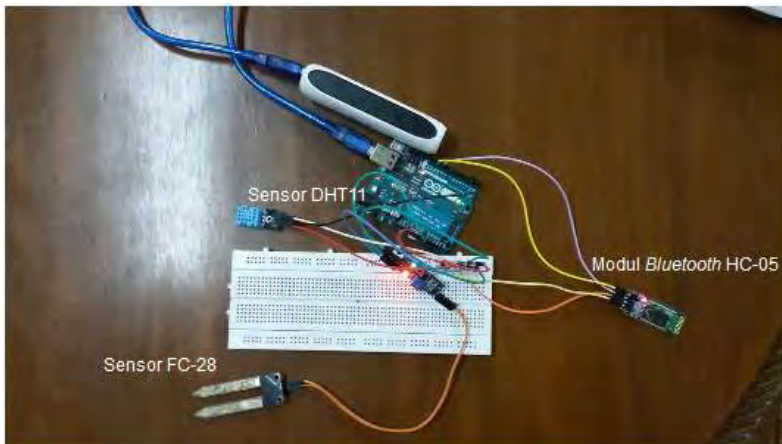
Tabel 4.3. Perangkat keras yang dibutuhkan

Perangkat Keras		Jumlah
Mikrokontroler	Arduino Uno Rev 3	Dua buah
	Arduino Mega 2560	Satu buah
Sensor	Sensor kelembaban tanah FC-28	Dua buah
	Sensor suhu dan kelembaban udara DHT11	Dua buah
Shield	GPS/GSM/GPRS <i>Shield</i> V3.0	Satu buah
Kabel	Kabel USB	Tiga buah
	Kabel <i>jumper</i>	Satu set
	Kabel adaptor	Satu buah
Board	<i>Breadboard</i>	Dua buah
Baterai	<i>Power bank</i> 5000mAh	Dua buah
Modul	<i>Bluetooth Module</i> HC-05	Empat buah

Perangkat keras yang dibutuhkan dalam system yang ditunjukkan pada Tabel 4.3 masing-masing memiliki perannya sendiri-sendiri. Perangkat keras dalam sistem ini akan diimplementasikan pada *slave node* dan *master node* sebagaimana dijelaskan pada Bab 4.2.1 dan Bab 4.2.2.

4.2.1 Implementasi Perangkat Keras pada *Slave Node*

Slave node bertugas untuk mengumpulkan data kondisi kelembaban tanah dan kelembaban udara pada lahan yang dipantau. Satu unit *slave node* terdiri dari satu buah mikrokontroler Arduino Uno yang diintegrasikan dengan sebuah sensor FC-28 yang berfungsi untuk mendapatkan data nilai kelembaban tanah, sebuah sensor DHT11 yang berfungsi untuk mendapatkan data nilai kelembaban udara, serta sebuah *Bluetooth module* HC-05 yang berfungsi sebagai alat komunikasi antara *slave node* dan *master node*. Implementasi perangkat keras pada *slave node* dapat dilihat pada Gambar 4.1.

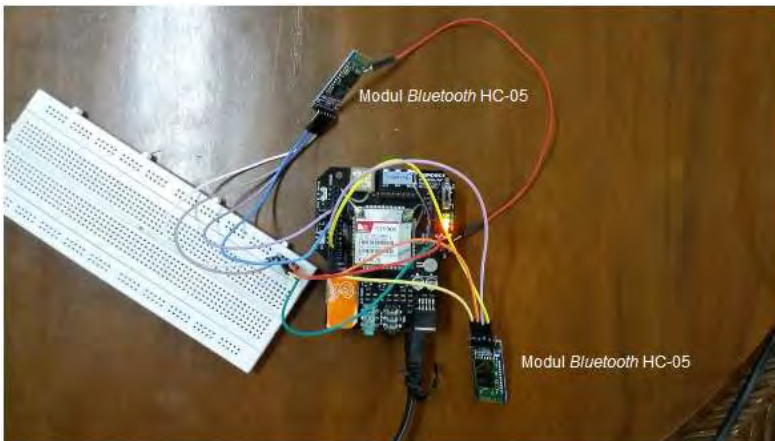


Gambar 4.1. Implementasi perangkat keras pada *slave node*

Sumber daya dan tegangan Arduino Uno berasal dari *power bank* yang dihubungkan dengan kabel USB. Sensor FC-28, sensor DHT11, dan modul *Bluetooth* HC-05 diintegrasikan dengan mikrokontroler Arduino dengan menggunakan kabel *jumper*. Agar perangkat-perangkat keras tersebut mendapatkan daya dan tegangan yang dibutuhkan untuk dapat beroperasi, daya dan tegangan didistribusikan lewat *Breadboard*.

4.2.2 Implementasi Perangkat Keras pada *Master Node*

Master node bertugas untuk mengumpulkan data dari *slave node* yang dikirim lewat koneksi *Bluetooth* dan kemudian mengirim data status lahan menuju *server*. *Master node* terdiri dari satu mikrokontroler Arduino Uno yang diintegrasikan dengan GSM/GPS/GPRS *Shield* dan dihubungkan dengan dua buah modul *Bluetooth* HC-05. Implementasi perangkat keras pada *master node* dapat dilihat pada Gambar.



Gambar 4.2. Implementasi perangkat keras pada *master node*

Master node mendapatkan daya dan tegangan dari kabel adaptor. Tidak seperti *slave node*, *master node* harus mengirimkan data menuju *server* lewat GSM/GPS/GPRS *Shield* yang membutuhkan banyak daya dan tegangan, karenanya dibutuhkan kabel adaptor dengan tegangan 12V dan daya 1A.

4.3 Implementasi Perangkat Lunak

Implementasi pada perangkat lunak terbagi ke dalam tiga bagian, yaitu implementasi mikrokontroler Arduino, *server*, dan aplikasi web. Setiap bagian akan dijelaskan lebih lanjut pada masing-masing subbab.

4.3.1 Implementasi Perangkat Lunak pada Arduino

Implementasi perangkat lunak pada Arduino terdiri dari implementasi perangkat lunak pada *slave node* dan implementasi perangkat lunak pada *master node*. Setiap bagian akan dijelaskan lebih lanjut pada masing-masing subbab.

4.3.1.1 Implementasi Perangkat Lunak pada *Slave Node*

Slave node bertugas untuk mengumpulkan nilai kelembaban tanah, suhu, dan kelembaban udara. *Slave node* juga bertugas untuk mengirim data tersebut kepada *master node*. Fungsi yang diimplementasikan adalah fungsi *setup* dan fungsi *loop*.

1	Set baud rate to 9600
2	Initialize <i>Bluetooth</i> pin
3	Initialize soilhumid sensor pin
4	Initialize humid sensor pin
5	Initialize <i>fuzzy logic</i>

Pseudocode 4.1. Fungsi setup pada slave node

Fungsi *setup* hanya dijalankan sekali ketika Arduino dinyalakan. Fungsi *setup* menginisialisasi *baud rate* atau jumlah bit data yang ditransmisikan per detik, menginisialisasi pin yang digunakan untuk modul *Bluetooth*, menginisialisasi pin yang digunakan untuk sensor yang digunakan, dan mengimplementasi *fuzzy logic*.

Inisialisasi untuk pin yang digunakan oleh modul *Bluetooth*, begitu juga dengan inisialisasi pin yang digunakan untuk sensor tidak memerlukan bentuk *logic* tersendiri, sementara itu untuk mengimplementasikan *fuzzy logic* memerlukan sebuah *logic* tersendiri yang dapat ditulis dalam bentuk *Pseudocode*.

Implementasi *fuzzy logic* terdiri dari menentukan *linguistic variable* dan menentukan *fuzzy rule*. Implementasi untuk menentukan *linguistic variable* dalam *fuzzy logic* dapat dilihat pada *Pseudocode 4.2*.

1	Set SoilHumid = new <i>FuzzyInput</i>
2	Add <i>InputLinguistic</i> (Kering)
3	Add <i>InputLinguistic</i> (Baik)
4	Add <i>InputLinguistic</i> (Lembab)
5	Set Humid = new <i>FuzzyInput</i>
6	Add <i>InputLinguistic</i> (Kering)
7	Add <i>InputLinguistic</i> (Baik)
8	Add <i>InputLinguistic</i> (Lembab)
9	Set Status = new <i>FuzzyOutput</i>
10	Add <i>OutputLinguistic</i> (SangatKering)
11	Add <i>OutputLinguistic</i> (ButuhPengairan)
12	Add <i>OutputLinguistic</i> (Baik)
13	Add <i>OutputLinguistic</i> (ButuhVentilasi)
14	Add <i>OutputLinguistic</i> (SangatLembab)

Pseudocode 4.2. Menentukan linguistic variables

Linguistic variable merupakan *variable* yang digunakan sebagai *input* dan *output* dalam *fuzzy logic*. Setelah mendefinisikan *linguistic variable*, yang dilakukan selanjutnya adalah menentukan *fuzzy rule*. *Fuzzy rule* digunakan untuk menentukan keluaran dari *fuzzy input*. Implementasi untuk menentukan *fuzzy rule* dalam *fuzzy logic* dapat dilihat pada *Pseudocode 4.3*.

1	Set Antecedent= new <i>FuzzyAntecedent</i>
2	Antecedent = joinWithAnd(SoilHumid,Humid)
3	Add Consequent = new <i>FuzzyConsequent</i>
4	<i>FuzzyRule</i> = (<i>RuleID</i> ,Antecedent,Consequent)

Pseudocode 4.3. Menentukan fuzzy rule

Dalam menentukan *fuzzy rule*, terdapat tiga bagian yang harus diperhatikan, yaitu antecedent, consequent, dan *fuzzy rule*. Antecedent adalah kondisi syarat *fuzzy input*, bisa dianalogikan dengan kondisi “If”. Consequent adalah kondisi yang merupakan konsekuensi dari antecedent, bisa dianalogikan dengan kondisi “Then”. *Fuzzy rule* adalah aturan dari metode *fuzzy logic* yang merupakan kombinasi antara antecedent dan consequent.

Pada fungsi *loop*, *slave node* melakukan pengumpulan data kondisi lahan yang didapat dari sensor FC-28 dan sensor DHT11. Pengumpulan data dilakukan dengan cara membaca nilai

kelembaban tanah dan kelembaban udara selama sepuluh detik, kemudian dirata-rata. Hasil dari rata-rata pembacaan tersebut kemudian menjadi *fuzzy input*. Setelah melewati proses defuzzifikasi dan menghasilkan *fuzzy output*, *output* tersebut kemudian dibandingkan selisihnya dengan *output* sebelumnya. Apabila hasil perbandingannya menunjukkan perbedaan kondisi yang signifikan, *slave node* akan melakukan pengiriman data menuju *master node* lewat modul *Bluetooth*. Implementasi fungsi *loop* pada *slave node* dapat dilihat pada *Pseudocode 4.4*.

1	TotalSoilHumid = 0
2	TotalHumid = 0
3	for i ← 1 to 10
4	SoilHumid[i]=read(soilhumid pin)
5	Humid[i]=read(humid pin)
6	TotalSoilHumid=TotalSoilHumid+SoilHumid[i]
7	TotalHumid=TotalHumid+Humid[i]
8	AvgSoilHumid=TotalSoilHumid/10
9	AvgHumid=TotalHumid/10
10	Fuzzify (AvgSoilHumid,AvgHumid)
11	Defuzzify()
12	Get FuzzyOutput
14	Flag=Compare (FuzzyOutput)
15	If flag > 2
16	Send(FuzzyOutput)

Pseudocode 4.4. Implementasi fungsi loop pada slave node

Fuzzy output dari proses defuzzifikasi berupa nilai status, yang masing-masing nilai menunjukkan kondisi lahan. Karena itu data *fuzzy output* dibandingkan lewat melihat selisih dengan *fuzzy output* dari proses sebelumnya. Jika memang terjadi perubahan maka data akan dikirimkan dengan sebuah variabel bertipe data char yang menunjukkan status kondisi lahan yang dipantau. Pengiriman dilakukan sedemikian rupa untuk memudahkan proses pengiriman menuju *server node* lewat *Bluetooth* yang menerima data dalam bentuk desimal. Data yang diterima dalam bentuk decimal selanjutnya akan dikonversi kembali menjadi status kondisi lahan pada *master node*.

4.3.1.2 Implementasi Perangkat Lunak pada *Master Node*

Master node bertugas untuk mengumpulkan nilai status lahan yang didapat dari *slave node* lewat paket yang dikirimkan via modul *Bluetooth*. Selain itu, *master node* juga bertugas untuk mengirimkan data status kondisi lahan menuju *server* secara adaptif dengan menggunakan koneksi internet yang diimplementasikan dengan perangkat *GSM/GPS/GPRS Shield*. Pada *master node*, Fungsi yang diimplementasikan adalah fungsi *setup* dan fungsi *loop*.

Fungsi *setup* pada *master node* dapat dilihat pada *Pseudocode 4.5*.

1	enable GSM mode
2	register_gsm()
3	Set baud rate to 9600
4	Initialize <i>Bluetooth1</i> pin
5	Initialize <i>Bluetooth2</i> pin
6	Set <i>Bluetooth1</i> rate to 9600
7	Set <i>Bluetooth2</i> rate to 9600
8	Initialize <i>fuzzy logic</i>

Pseudocode 4.5. Fungsi setup pada master node

Pada fungsi *setup*, *master node* menginisialisasi pin yang digunakan untuk dua buah modul *Bluetooth*. Inisialisasi *Bluetooth* dapat langsung dilakukan begitu saja karena berbeda dengan mikrokontroler Arduino Uno yang digunakan sebagai *slave node* yang hanya memiliki sepasang pin komunikasi, mikrokontroler Arduino Mega 2560 memiliki empat pasang pin untuk komunikasi sehingga masing-masing komunikasi serial yang dilakukan oleh *Bluetooth module* tidak perlu terjadi *interrupt*. Baud rate pada masing-masing modul *Bluetooth* pun didefinisikan di fungsi *setup* agar modul dapat berkomunikasi satu sama lain. Inisialisasi modul *Bluetooth* pada *master node* sangat penting karena untuk itulah tugas sebuah *node* yang berperan sebagai *gateway*, yaitu untuk berkomunikasi dan mengumpulkan data dari *node-node* yang tersebar pada sebuah jaringan sensor nirkabel.

Selain itu, fungsi *setup* juga meregister sim card GSM sehingga dapat digunakan untuk mengakses internet. Di dalam

fungsi *setup* pula *fuzzy logic* untuk menentukan kapan pengiriman data menuju *server* diinisialisasi.

Inisialisasi *fuzzy logic* pada *master node* dapat dilihat pada *Pseudocode 4.6*.

1	Set Status1 = new <i>FuzzyInput</i>
2	Add <i>InputLinguistic</i> (SangatKering)
3	Add <i>InputLinguistic</i> (ButuhPengairan)
4	Add <i>InputLinguistic</i> (Baik)
5	Add <i>InputLinguistic</i> (ButuhVentilasi)
6	Add <i>InputLinguistic</i> (SangatLembab)
7	Set Status2 = new <i>FuzzyInput</i>
8	Add <i>InputLinguistic</i> (SangatKering)
9	Add <i>InputLinguistic</i> (ButuhPengairan)
10	Add <i>InputLinguistic</i> (Baik)
11	Add <i>InputLinguistic</i> (ButuhVentilasi)
12	Add <i>InputLinguistic</i> (SangatLembab)
13	Set StatusAkhir = new <i>FuzzyOutput</i>
14	Add <i>OutputLinguistic</i> (SangatKering)
15	Add <i>OutputLinguistic</i> (ButuhPengairan)
16	Add <i>OutputLinguistic</i> (Baik)
17	Add <i>OutputLinguistic</i> (ButuhVentilasi)
18	Add <i>OutputLinguistic</i> (SangatLembab)

Pseudocode 4.6. Inisialisasi fuzzy logic pada master node

Fuzzy input pada *master node* terdiri dari dua nilai, yaitu kondisi lahan hasil pemantauan dari dua *slave node*. Setelah menentukan variabel untuk *linguistic variable*, yang dilakukan selanjutnya adalah menentukan *fuzzy rule*. Inisialisasi *fuzzy rule* pada *master node* dapat dilihat pada *Pseudocode 4.7*.

1	Set Antecedent= new <i>FuzzyAntecedent</i>
2	Antecedent = joinWithAnd(Status1, Status2)
3	Add Consequent = new <i>FuzzyConsequent</i>
4	<i>FuzzyRule</i> = (<i>RuleID</i> , Antecedent, Consequent)

Pseudocode 4.7. Inisialisasi fuzzy rule pada master node

Selanjutnya, *master node* akan mengirimkan data lewat GSM/GPRS/GPS *Shield* dengan menggunakan fungsi *SendData()* yang *Pseudocode*-nya dapat dilihat pada *Pseudocode 4.8*

1	<i>Command</i> (“AT+HTTPIPINIT”)
2	<i>Command</i> (“AT+HTTTPARA=CID,1”)
3	<i>Command</i> (“AT+HTTTPARA=URL”)
4	//set http parameter value
5	<i>Command</i> (“AT+HTTTPACTION”)
6	//set HTTP method
7	

Pseudocode 4.8. Fungsi SendData

Fungsi *SendData()* merupakan fungsi yang dijalankan dengan *AT Command* milik SIM900, yang tentunya berbeda dengan *AT Command* yang dimiliki oleh modul *Bluetooth* HC-05.

Fungsi *loop* pada *master node* menjalankan fungsi menerima data dari kedua *slave node*. Data-data yang telah diterima Rata-rata tersebut kemudian menjadi *fuzzy input* untuk menentukan apakah data akan dikirimkan menuju *server* atau tidak. Fungsi *loop* pada *master node* dapat dilihat pada *Pseudocode 4.8*.

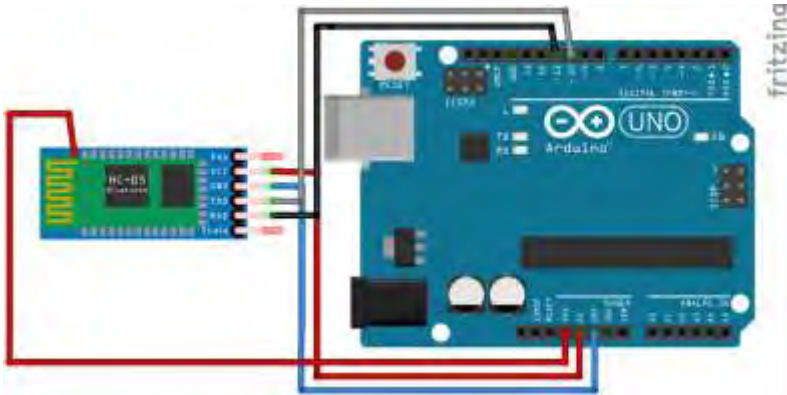
1	if <i>Bluetooth1.available()</i> OR
2	<i>Bluetooth2.available()</i>
3	<i>data1</i> = <i>read(Bluetooth1)</i>
4	<i>data2</i> = <i>read(Bluetooth2)</i>
5	<i>Fuzzify(data1,data2)</i>
6	<i>Defuzzify()</i>
7	<i>Get FuzzyOutput</i>
8	<i>Flag=Compare(FuzzyOutput)</i>
9	If <i>flag</i> > 2
10	<i>SendData()</i>

Pseudocode 4.9. Fungsi loop pada master node

4.3.2 Implementasi Perangkat Lunak pada Modul *Bluetooth*

Pemrograman pada modul *Bluetooth* HC-05 dilakukan dengan aturan tersendiri yang disebut dengan *AT Command*. Untuk dapat melakukan pemrograman pada modul *Bluetooth*, harus masuk terlebih dahulu ke dalam *AT mode*. Untuk dapat masuk ke dalam *AT mode*, pin 34 pada *Bluetooth* HC-05 harus di-

setting high terlebih dahulu atau memberi tegangan pada pin 34 sebagaimana yang ditunjukkan pada Gambar 4.3.



Gambar 4.3. Rangkaian HC-05 dalam AT mode

AT *command* berguna untuk menentukan role dari modul *Bluetooth* apakah dia menjadi *master* ataukah *slave* dan untuk menentukan alamat yang akan dituju untuk pengiriman data. Implementasi pada modul *Bluetooth* ini dilakukan di *master node*, karena secara *default* modul *Bluetooth* HC-05 berperan sebagai *slave*. Karena itu untuk mengubah sebuah modul menjadi *master* maka dibutuhkan AT *Command*. AT *Command* ini hanya akan dilakukan sekali, yaitu ketika menginisialisasi modul *Bluetooth*.

Implementasi AT *command* pada *master node* dilakukan sekaligus, tetapi secara umum terdapat langkah-langkah yang membuat masing-masing *command* berbeda dari yang lain. Yaitu langkah inisialisasi, langkah inquiry, dan langkah *connecting*.

Langkah inisialisasi adalah langkah modul *Bluetooth* didefinisikan role-nya. Implementasi langkah inisialisasi modul *Bluetooth* dapat dilihat pada *Pseudocode* 4.9.

1	AT+RMAAD
2	AT+ROLE=1
3	AT+RESET
4	AT+CMODE=0

Pseudocode 4.10. Implementasi langkah inisialisasi modul *Bluetooth*

AT+RMAAD berguna untuk menghapus jejak hubungan yang mungkin sempat tersambung dengan modul *Bluetooth* lainnya. AT+ROLE=1 menginisialisasi peran modul *Bluetooth* menjadi *master*. AT+RESET untuk me-reset kembali modul *Bluetooth* setelah sebelumnya berganti role. AT+CMODE=0 menginisialisasi modul *Bluetooth* untuk dapat terkoneksi dengan perangkat apa saja.

Langkah inquiry adalah langkah modul *Bluetooth* mencari perangkat lain untuk dihubungkan. Implementasi langkah inquiry modul *Bluetooth* ini dapat dilihat pada *Pseudocode* 4.10.

1	AT+INQM=0, 5, 9
2	AT+INIT
3	AT+INQ

Pseudocode 4.11. Implementasi langkah inquiry modul Bluetooth

AT+INQM=0,5,9 memberikan modul *Bluetooth* perintah untuk mencari lima perangkat *Bluetooth* lain di sekitar modul dalam waktu 9 detik. AT+INIT berguna untuk mengaktifkan SPP profile pada modul *Bluetooth*. SPP profile berguna untuk memungkinkan adanya virtual serial port dan menghubungkan dua perangkat *Bluetooth*. AT+INQ untuk melihat list perangkat *Bluetooth* yang tersedia.

Langkah connecting adalah langkah di mana modul *Bluetooth* saling terhubung dengan modul *Bluetooth* yang lain. Implementasi langkah connecting pada modul *Bluetooth* ini dapat dilihat pada *Pseudocode* 4.11.

1	AT+PAIR= (ADDR) , (TIMEOUT)
2	AT+BIND= (ADDR)
3	AT+CMODE=1
4	AT+LINK= (ADDR)

Pseudocode 4.12. Implementasi langkah connecting pada modul Bluetooth

AT+PAIR adalah *command* yang berguna untuk memasang dua perangkat *Bluetooth*. Memerlukan dua parameter yaitu alamat perangkat yang ingin dipasang dan juga waktu timeout. Jika sampai waktu timeout lewat dan kedua

perangkat belum terpasangkan, maka pemasangan pertama belum berhasil. AT+BIND berguna untuk mengecek alamat dari perangkat yang sudah dipasangkan. AT+CMODE=1 menginisialisasi modul *Bluetooth* untuk hanya dapat terkoneksi dengan alamat atau perangkat yang telah sebelumnya terpasangkan. AT+LINK untuk menghubungkan kedua modul *Bluetooth*.

Dengan implementasi yang dijelaskan pada Bab 4.3.2 ini, *Bluetooth* akan dengan otomatis terkoneksi dengan perangkat *Bluetooth* lain yang telah diinisiasi. Dengan *command* inilah *master node* dapat terhubung langsung ke *slave node*.

(Halaman ini sengaja dikosongkan)

BAB V

PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, skenario uji coba yang meliputi uji tingkat akurasi dan uji kinerja serta analisa setiap pengujian.

5.1 Lingkungan Uji Coba

Lingkungan uji coba sistem pemantauan kondisi kelembaban lahan menggunakan bantuan perangkat keras dan perangkat lunak. Spesifikasi perangkat keras yang digunakan dalam pelaksanaan uji coba dapat dilihat pada Tabel 5.1.

Tabel 5.1. Spesifikasi perangkat keras dalam uji coba

Perangkat Keras	Spesifikasi	
Acer Aspire M3970	RAM	4GB
	Processor	Intel®Core™i3-2120 3.30GHz

Spesifikasi perangkat lunak yang digunakan dalam pelaksanaan uji coba sistem dapat dilihat pada Tabel 5.2.

Tabel 5.2. Spesifikasi perangkat lunak dalam uji coba

Sistem Operasi	Windows 8.1 64-bit
Arduino IDE	Arduino 1.6.7
Web Browser	Mozilla Firefox 46.0.1

Sementara itu untuk lingkungan lahan yang dipantau, terdapat dua buah *sample* tanah yang digunakan sebagai bahan uji.

Masing-masing tanah kondisinya berbeda sebagai bahan perbandingan dalam pengambilan data pada saat pengujian.



Gambar 5.1. Sample tanah dalam pengujian

5.2 Skenario Uji Coba

Uji coba ini dilakukan untuk menguji apakah fungsionalitas program telah diimplementasikan dengan benar dan berjalan sesuai rencana. Pada sistem ini, akan dilakukan uji coba fungsionalitas sistem dan uji coba performa dan tingkat akurasi sistem.

5.2.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas dilakukan untuk mengetahui kesesuaian keluaran dari tiap tahap atau langkah penggunaan fitur terhadap skenario yang dipersiapkan. Berikut ini penjabaran skenario dan hasil uji coba fungsionalitas yang dilakukan terhadap sistem.

5.2.1.1 Uji Coba Modul *Bluetooth* HC-05

Komunikasi antar *node* sangat bergantung pada koneksi *Bluetooth*, karena itu inisialisasi modul *Bluetooth* harus dilakukan terlebih dahulu.

Inisialisasi dilakukan hanya pada modul *Bluetooth* yang berperan sebagai *master node*. Inisialisasi dilakukan dengan memasukkan *AT command* lewat serial monitor pada Arduino IDE. Inisialisasi meliputi inisialisasi role atau peran untuk dapat menjadi *master node*, pencarian perangkat *Bluetooth* yang ingin dikoneksikan, hingga menghubungkan *master node* ke *slave node* yang dituju. List *AT command* yang dijalankan dapat dilihat pada Tabel 5.3.

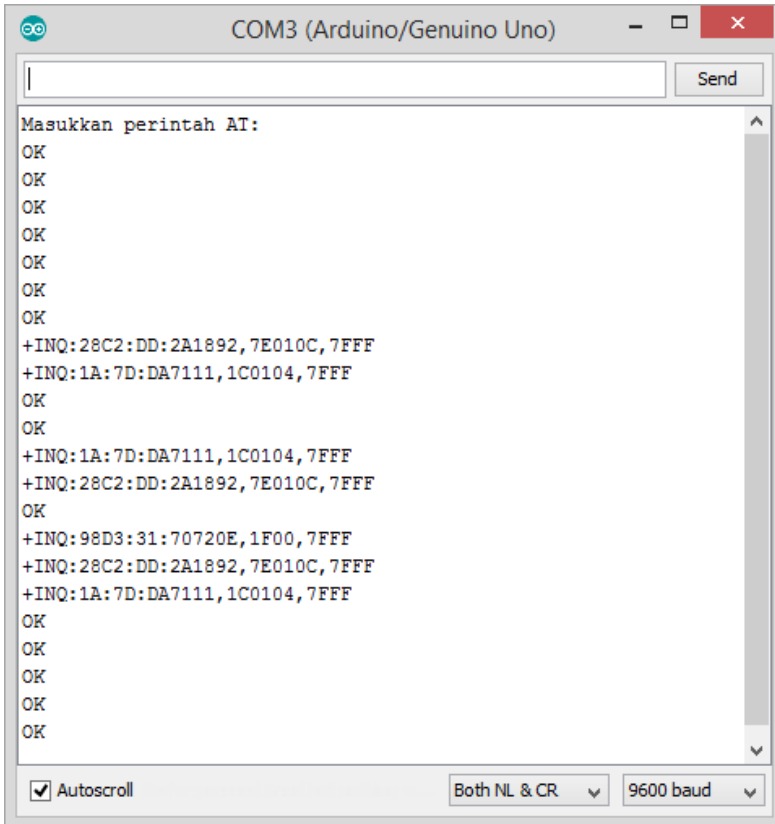
Tabel 5.3. AT Command pada master node

```
AT+RMAAD
AT+ROLE=1
AT+RESET
AT+CMODE=0
AT+INQM=0,5,9
AT+INIT
AT+INQ
AT+PAIR=(ADDR),(TIMEOUT)
AT+BIND=(ADDR)
AT+CMODE=1
AT+LINK=(ADDR)
```

Masing-masing *command* memiliki fungsinya masing-masing. “AT+RMAAD” berfungsi untuk menghapus koneksi dari perangkat yang tersambung sebelumnya. “AT+ROLE” untuk menentukan peran modul, apakah akan menjadi *master* atau menjadi *slave*. “AT+RESET” adalah perintah untuk me-reset modul. Perintah *reset* ini harus dilakukan setiap kali pengguna mengubah *role* dari modul *Bluetooth*. “AT+CMODE” untuk setting koneksi *Bluetooth*, apakah ia mampu terkoneksi dengan berbagai perangkat atau hanya terkoneksi dengan satu alamat terakhir yang tercatat pernah terkoneksi dengan modul *Bluetooth*. “AT+INIT” untuk menginisialisasi SPP atau *Serial Port Protocol* yang memungkinkan dua perangkat dapat terkoneksi. “AT+INQ” untuk mencari perangkat *Bluetooth*. “AT+PAIR” untuk mencoba mengkoneksikan dua modul *Bluetooth* dengan waktu *timeout* tertentu. “AT+BIND” untuk menghubungkan dua modul *Bluetooth*. Dan terakhir, “AT+LINK” untuk mengkoneksikan dua

perangkat *Bluetooth* secara permanen sehingga ketika kedua perangkat tersebut menyala, akan secara otomatis mencari alamat terakhir dari perangkat yang terkoneksi sebelumnya.

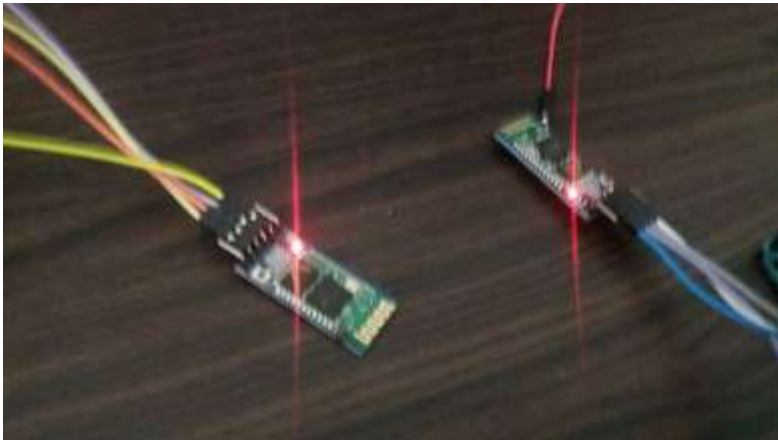
Respon dari perintah tersebut dapat dilihat pada Gambar 5.2.



Gambar 5.2. Respon terhadap AT Command

Sebagaimana yang ditunjukkan pada Gambar 5.2, respon modul *Bluetooth* terhadap AT Command yang diberikan muncul di serial monitor. Respon AT Command banyak yang hanya konfirmasi berupa “OK” yang artinya inisialisasi atau AT

command yang diberikan sukses. Sementara itu, respon “+INQ” adalah respon dari perintah “AT+INQ” yang mana perintah tersebut adalah perintah untuk mencari perangkat *Bluetooth* di sekitarnya. Modul *Bluetooth* merespon dengan mengirimkan alamat perangkat *Bluetooth* lain yang diterima oleh modul *Bluetooth*. Setelah menerima alamat perangkat *Bluetooth* lain, modul *Bluetooth* akan melakukan pengkoneksian dengan perangkat yang dituju. Jika kedua perangkat tersebut telah terhubung, maka lampu indikator pada kedua modul akan berkedip bersamaan sebagaimana yang ditunjukkan pada Gambar 5.3.



Gambar 5.3. Dua modul *Bluetooth* yang telah terhubung

5.2.1.2 Uji Coba Mendeteksi Nilai Sensor dan *Fuzzy logic* pada *Slave Node*

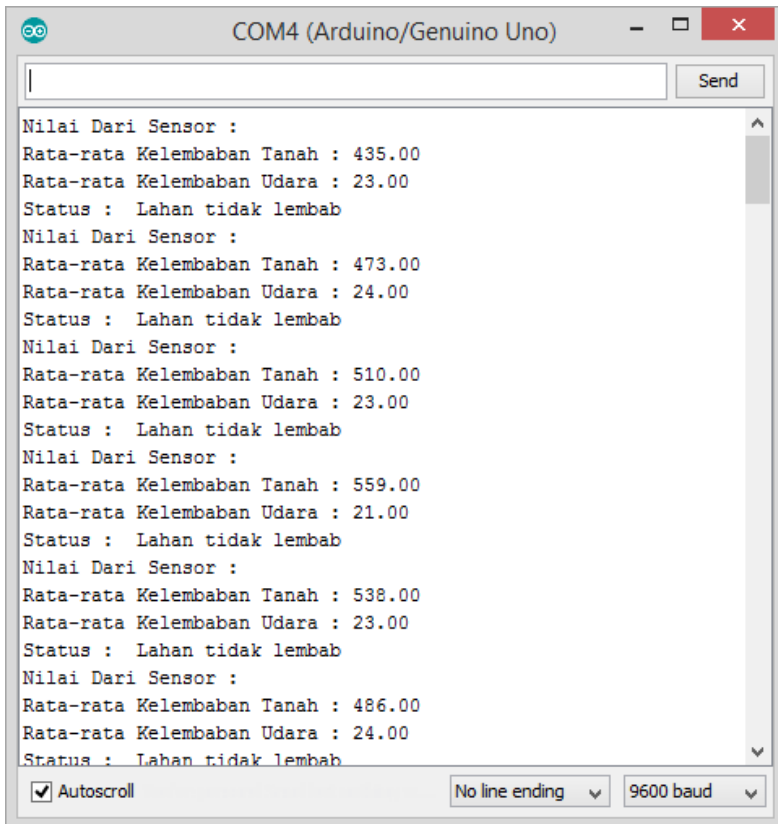
Slave node bertugas untuk melakukan pembacaan data yang didapat dari dua jenis sensor, yaitu sensor kelembaban tanah FC-28 dan sensor kelembaban udara DHT11. Selain itu, *slave node* juga mengimplementasi metode *fuzzy logic* untuk menentukan status lahan yang dipantau. Bab ini menjelaskan uji coba fungsionalitas *slave node* tersebut.

Seperti yang telah dijelaskan pada Bab 4, *slave node* terdiri dari mikrokontroler Arduino yang dirangkai dengan sensor FC-28, sensor DHT11, dan modul *Bluetooth* HC-05. Rangkaian tersebut diuji dengan tanah sebagai *sample* lahan yang dipantau. Kegiatan pengujian tersebut dapat dilihat pada Gambar 5.4.



Gambar 5.4. Uji coba rangkaian *slave node*

Sensor FC-28 ditancapkan ke *sample* tanah untuk melakukan pengujian, sedangkan sensor DHT11 dibiarkan terbuka begitu saja. Masing-masing sensor membaca data nilai kelembaban, yaitu data nilai kelembaban tanah dan data nilai kelembaban udara. Data hasil pembacaan sensor tersebut kemudian dikumpulkan selama waktu *delay* yang ditentukan, dalam pengujian ini dilakukan setiap sepuluh detik. Selanjutnya data yang telah dikumpulkan tadi dirata-rata dan diintegrasikan dengan menggunakan metode *fuzzy logic* untuk menghasilkan status kelembaban dari lahan yang dipantau. Status kelembaban lahan inilah yang nantinya akan dikirimkan menuju *master node* dengan menggunakan koneksi *Bluetooth*. Hasil pengujian tersebut dapat dilihat pada Gambar 5.5.

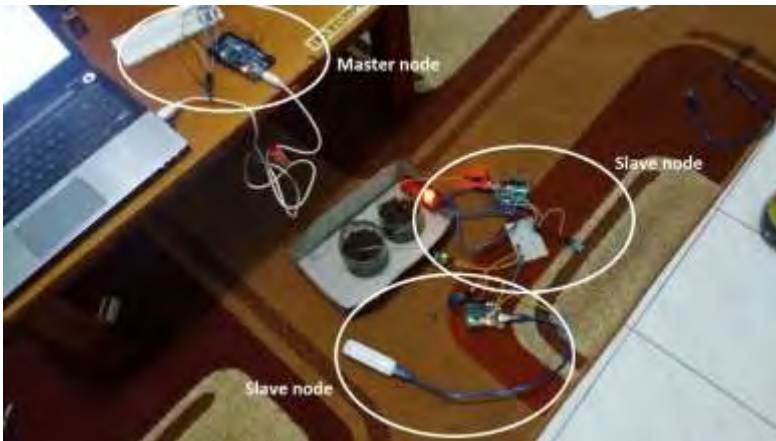


Gambar 5.5. Uji coba pada *slave node*

5.2.1.3 Uji Coba Pengiriman Data dari *Slave Node* Menuju *Master Node*

Pengiriman data menuju *master node* yang dilakukan oleh *slave node* dengan menggunakan modul *Bluetooth* HC-05 dilakukan hanya ketika hasil agregasi data dari metode *fuzzy logic* menunjukkan adanya perubahan status kelembaban lahan.

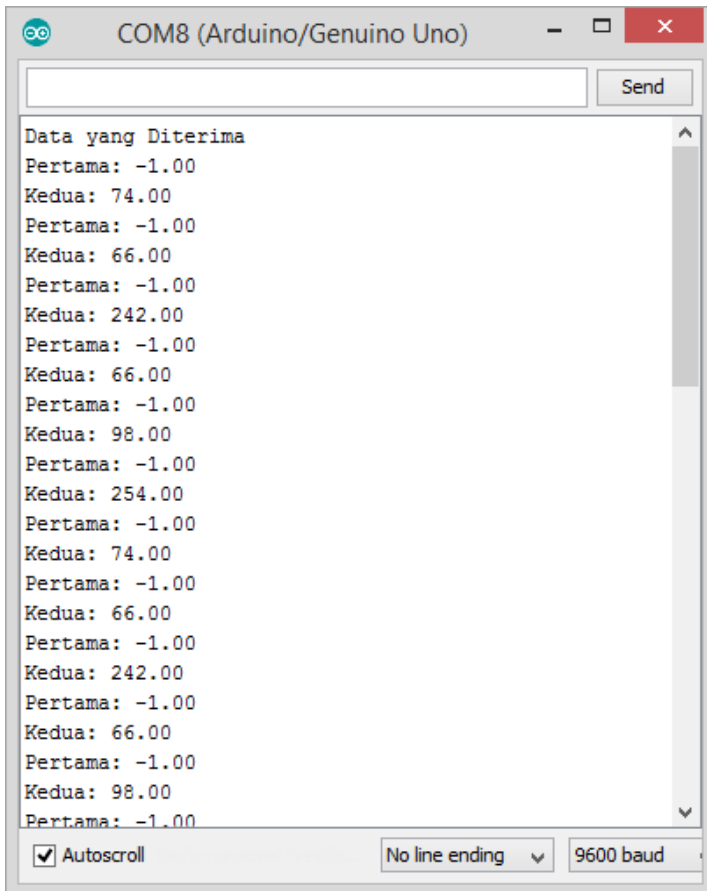
Uji coba rangkaian *slave node* dan *master node* yang terhubung dengan menggunakan koneksi *Bluetooth* dapat dilihat pada Gambar 5.6.



Gambar 5.6. Rangkaian *master node* dan *slave node*

Rangkaian dua unit *slave node* yang terhubung dengan satu unit *master node* dengan menggunakan koneksi *Bluetooth* sudah bias dikatakan sebagai sebuah jaringan sensor nirkabel berbasis *Bluetooth*. Pada perangkat yang berperan sebagai *Slave node*, masing-masingnya dibangun oleh satu buah mikrokontroler Arduino Uno, satu buah sensor DHT11, satu buah sensor FC-28, dan satu buah modul *Bluetooth* HC-05. Sementara itu perangkat yang berperan sebagai *master node* dibangun oleh satu buah mikrokontroler Arduino Mega 2560 dan dua buah modul *Bluetooth* HC-05.

Modul *Bluetooth* HC-05 mengirimkan datanya kepada *node* lain ke dalam bentuk *decimal*. Sehingga status yang dikirimkan oleh *slave node* harus dikonversikan terlebih dahulu ke dalam bentuk *char* agar lebih mudah untuk kemudian nantinya dikonversikan kembali. Hasil uji coba pengiriman data dari *slave node* menuju *master node* dapat dilihat pada Gambar 5.7

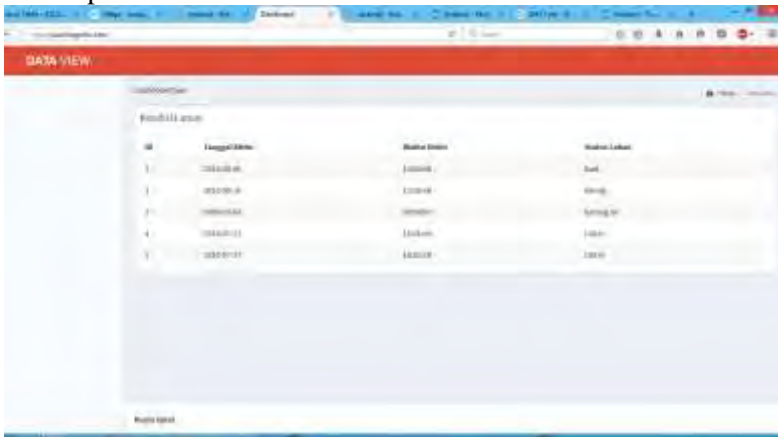


Gambar 5.7. Uji coba pengiriman data ke *master node*

5.2.1.4 Uji Coba Pengiriman Menuju *Server*

Pengiriman data menuju *server* dilakukan oleh *master node* dengan menggunakan *GSM/GPS/GPRS Shield*. Yang dikirimkan oleh *master node* adalah kondisi kelembaban lahan terakhir. Sehingga dengan ini yang tampak adalah perubahan dari

kondisi lahan terakhir. Sehingga pengguna tidak perlu terus menerus memantau lahannya, cukup dengan melihat data yang ditampilkan oleh aplikasi maka akan terlihat kondisi terakhir dari lahannya. Karena apabila ada perubahan maka data akan langsung ter-update, sedangkan jika tidak terdapat perubahan maka tidak terjadi apa-apa. Hasil uji coba pengiriman data ke *server* dapat dilihat pada



ID	Tanggal/Minggu	Mudra Suhu	Mudra Lahan
1	2024-09-01	25.000	Sedang
2	2024-09-02	25.000	Sedang
3	2024-09-03	25.000	Sedang
4	2024-09-04	25.000	Sedang
5	2024-09-05	25.000	Sedang

Gambar 5.8. Hasil uji coba pengiriman ke *server*

5.2.2 Uji Coba Performa

Selain uji coba fungsionalitas, performa dari sistem pemantauan kondisi lahan pun perlu diuji untuk membuktikan apakah sistem yang bekerja secara adaptif mampu memiliki performa yang lebih baik dari segi penghematan baterai.

5.2.2.1 Uji Coba Performa Baterai pada Sistem

Jalannya sistem sangat bergantung pada baterai mengingat *slave node* yang merupakan komponen utama yang melakukan pengumpulan data diberi daya oleh baterai. Pada subbab ini akan dijelaskan mengenai pengujian baterai pada sistem yang

mengadopsi penjadwalan adaptif dengan kondisi bila sistem tidak terjadwal secara adaptif.

Baterai yang digunakan adalah sebuah *power bank* yang memiliki spesifikasi sebagaimana tertera pada Tabel 5.4.

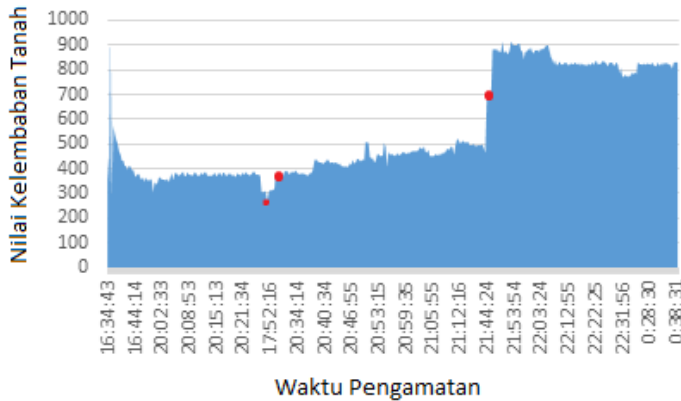
Tabel 5.4. Spesifikasi *power bank*

Model	Samsung A028
Kapasitas	5000mAh
Input	DC 0.5V-1000mA
Output	DC 5.3V 1500mA

Pada *node* yang tidak diimplementasikan sifat adaptif, yaitu *node* langsung mengirimkan hasil pembacaan datanya, baterai tahan selama kurang lebih 6 jam. Sementara itu untuk *node* yang diimplementasikan sifat adaptif, baterai tahan selama kurang lebih 6,5 jam. Sehingga terbukti sifat adaptif yang diimplementasikan lebih tahan 8,3 persen.

5.3 Analisis Hasil Uji Coba

Dari hasil uji coba yang telah dilakukan, ada beberapa hal yang dapat dianalisis. Salah satunya adalah sifat adaptif yang lebih menghemat baterai, meski dengan penghematan yang tidak sampai 10%. Hal ini dapat disebabkan karena meskipun *node* tidak mengirimkan data, namun selama *node* tersambung dengan sebuah perangkat maka akan tetap mengkonsumsi banyak daya. Padahal dengan sifat adaptif, pengiriman data dapat dilakukan hanya beberapa kali saja dan 24 jam. Karena kondisi lahan adalah kondisi yang relatif statis, tidak mudah berubah. Perubahannya dalam beberapa jam menunjukkan hasil yang tidak signifikan. Contohnya pada data kelembaban tanah yang diambil pada bulan April lalu. Selama delapan jam pengambilan dan pengamatan data, tercatat hanya ada tiga kali perubahan status kondisi kelembaban tanah. Hal ini dapat dilihat pada grafik sebagaimana yang ditunjukkan pada Gambar 5.9



Gambar 5.9. Grafik kondisi kelembaban tanah

Begitu pun halnya dngan kelembaban udara. Mendapatkan perubahan kelembaban udara cukup sulit mengingat tergantung pada cuaca. Apabila tidak ada perubahan drastic maka tidak akan bergeser status kelembaban udara itu. Selama 24 jam pengamatan, tidak ada perubahan status kelembaban udara sebagaimana ditunjukkan grafik pada Gambar 5.10.



Gambar 5.10. Grafik kondisi kelembaban udara [11]

LAMPIRAN

1. Inisialisasi *fuzzy logic* pada *slave node*.

```
Fuzzy* fuzzy = new Fuzzy();

FuzzySet* tanahlembab = new FuzzySet(0, 100, 300, 400);
FuzzySet* tanahbaik = new FuzzySet(300, 415, 635, 750);
FuzzySet* tanahkering = new FuzzySet(700, 800, 900, 1000);

FuzzySet* udarakering = new FuzzySet(0, 15, 30, 45);
FuzzySet* udarabaik = new FuzzySet(40, 50, 60, 70);
FuzzySet* udaralembab = new FuzzySet(65, 75, 90, 100);

void setup() {
    Serial.begin(9600);
    BTserial.begin(9600);

    // initialize all the readings to 0:
    for (int thisReading = 0; thisReading < numReadings;
thisReading++) {
        readings[thisReading] = 0;
    }

    // FuzzyInput
    FuzzyInput* kelembabantanah = new FuzzyInput(1);
    kelembabantanah->addFuzzySet(tanahkering);
    kelembabantanah->addFuzzySet(tanahbaik);
    kelembabantanah->addFuzzySet(tanahlembab);

    fuzzy->addFuzzyInput(kelembabantanah);

    // FuzzyInput
    FuzzyInput* kelembabanudara = new FuzzyInput(2);
    kelembabanudara->addFuzzySet(udarakering);
    kelembabanudara->addFuzzySet(udarabaik);
    kelembabanudara->addFuzzySet(udaralembab);

    fuzzy->addFuzzyInput(kelembabanudara);

    // FuzzyOutput
    FuzzyOutput* kelembabanlahan = new FuzzyOutput(1);
```

```

FuzzyOutput* kelembabanlahan = new FuzzyOutput(1);

FuzzySet* sangatkering = new FuzzySet(0, 5, 15, 20);
kelembabanlahan->addFuzzySet(sangatkering);
FuzzySet* butuhpengairan = new FuzzySet(20, 25, 35, 40);
kelembabanlahan->addFuzzySet(butuhpengairan);
FuzzySet* baik = new FuzzySet(40, 45, 55, 60);
kelembabanlahan->addFuzzySet(baik);
FuzzySet* butuhventilasi = new FuzzySet(60, 65, 75, 80);
kelembabanlahan->addFuzzySet(butuhventilasi);
FuzzySet* sangatlembab = new FuzzySet(80, 85, 95, 100);
kelembabanlahan->addFuzzySet(sangatlembab);

fuzzy->addFuzzyOutput(kelembabanlahan);

// Building FuzzyRule
FuzzyRuleAntecedent* tanahkeringudarakering = new
FuzzyRuleAntecedent();
tanahkeringudarakering->joinWithAND(tanahkering,
udarakering);

FuzzyRuleConsequent* makalahansangatkering = new
FuzzyRuleConsequent();
makalahansangatkering->addOutput(sangatkering);

FuzzyRule* fuzzyRule1 = new FuzzyRule(1,
tanahkeringudarakering, makalahansangatkering);
fuzzy->addFuzzyRule(fuzzyRule1);

// Building FuzzyRule
FuzzyRuleAntecedent* tanahkeringudarabaik = new
FuzzyRuleAntecedent();
tanahkeringudarabaik->joinWithAND(tanahkering,
udarabaik);
FuzzyRuleAntecedent* tanahbaikudarakering = new
FuzzyRuleAntecedent();
tanahbaikudarakering->joinWithAND(tanahbaik,

```

```

    tanahbaikudarakering->joinWithAND(tanahbaik,
    udarakering);

    FuzzyRuleAntecedent*
    tanahkeringudarabaikortanahbaikudarakering      =      new
    FuzzyRuleAntecedent();

    tanahkeringudarabaikortanahbaikudarakering->joinWithOR(t
    anahbaikudarakering, tanahkeringudarabaik);

    FuzzyRuleConsequent* makalahanbutuhpengairan = new
    FuzzyRuleConsequent();
    makalahanbutuhpengairan->addOutput(butuhpengairan);

    FuzzyRule* fuzzyRule2 = new FuzzyRule(2,
    tanahkeringudarabaikortanahbaikudarakering,
    makalahanbutuhpengairan);
    fuzzy->addFuzzyRule(fuzzyRule2);

    // Building FuzzyRule
    FuzzyRuleAntecedent* tanahbaikudarabaik = new
    FuzzyRuleAntecedent();
    tanahbaikudarabaik->joinWithAND(tanahbaik, udarabaik);

    FuzzyRuleConsequent* makalahanbaik = new
    FuzzyRuleConsequent();
    makalahanbaik->addOutput(baik);

    FuzzyRule* fuzzyRule3 = new FuzzyRule(3,
    tanahbaikudarabaik, makalahanbaik);
    fuzzy->addFuzzyRule(fuzzyRule3);

    // Building FuzzyRule
    FuzzyRuleAntecedent* tanahbaikudaralembab = new
    FuzzyRuleAntecedent();
    tanahbaikudaralembab->joinWithAND(tanahbaik,
    udaralembab);
    FuzzyRuleAntecedent* tanahlembabudarabaik = new

```

```

    FuzzyRuleAntecedent* tanahlembabudarabaik = new
    FuzzyRuleAntecedent();
    tanahlembabudarabaik->joinWithAND(tanahlembab,
    udarabaik);

    FuzzyRuleAntecedent*
    tanahbaikudaralembababortanahlembabudarabaik = new
    FuzzyRuleAntecedent();

    tanahbaikudaralembababortanahlembabudarabaik->joinWithOR(t
    anahlembabudarabaik, tanahbaikudaralembab);

    FuzzyRuleConsequent* makalahanbutuhventilasi = new
    FuzzyRuleConsequent();
    makalahanbutuhventilasi->addOutput(butuhventilasi);

    FuzzyRule* fuzzyRule4 = new FuzzyRule(4,
    tanahbaikudaralembababortanahlembabudarabaik,
    makalahanbutuhventilasi);
    fuzzy->addFuzzyRule(fuzzyRule4);

    // Building FuzzyRule
    FuzzyRuleAntecedent* tanahlembabudaralembab = new
    FuzzyRuleAntecedent();
    tanahlembabudaralembab->joinWithAND(tanahlembab,
    udaralembab);

    FuzzyRuleConsequent* makalahansangatlembab = new
    FuzzyRuleConsequent();
    makalahansangatlembab->addOutput(sangatlembab);

    FuzzyRule* fuzzyRule5 = new FuzzyRule(5,
    tanahlembabudaralembab, makalahansangatlembab);
    fuzzy->addFuzzyRule(fuzzyRule5);
}

```

BAB VI

KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dari hasil pengamatan dan percobaan selama perancangan, implementasi, dan uji coba aplikasi, maka dapat diambil kesimpulan sebagai berikut:

1. Untuk merancang jaringan sensor nirkabel berbasis *Bluetooth*, yang perlu diperhatikan adalah menentukan peran-peran pada masing-masing *node*.
2. Untuk melakukan pengiriman data, kondisi yang sebelumnya perlu diperhatikan adalah tentang kapan melakukan pengiriman data tersebut
3. Implementasi metode *fuzzy* dapat dilakukan sebagai solusi untuk menentukan jadwal pengiriman data secara adaptif
4. Hasil pengujian pengiriman data, dari 4 kali percobaan, 3 kali berhasil. Hal ini disebabkan karena *delay* antara pengiriman data dengan Arduino
5. Sifat adaptif yang diimplementasikan dapat membuat baterai lebih tahan lama 8,3 persen.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang, berdasarkan pada hasil perancangan, implementasi dan uji coba yang telah dilakukan.

1. Menggunakan teknologi dan perangkat yang lebih canggih, seperti mengganti *Bluetooth module* dengan *Bluetooth shield*. Karena keterbatasan perangkat keras sedikit banyak

memengaruhi kerja sistem. *Bluetooth module* memiliki banyak kekurangan, diantaranya jangkauan sinyal yang pendek membuat lingkup pemantauan lahan menjadi sempit. Selain itu, dengan mengganti *Bluetooth module* dengan *Bluetooth shield*, fungsi yang diimplementasikan akan lebih banyak. Dengan menggunakan *Bluetooth shield*, teknologi *Bluetooth* seperti piconet dapat diimplementasikan sehingga satu *master node* dapat terhubung dengan hingga tujuh *slave node* sekaligus.

2. Untuk meningkatkan akurasi dan kualitas data kondisi lahan yang diambil, ada baiknya bila sensor yang digunakan ditambah. Misal sensor pH tanah.

DAFTAR PUSTAKA

- [1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister, "System Architecture Directions for Networked Sensors," in *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Massachusetts, 2000.
- [2] S. Krco, "Bluetooth Based Wireless Sensor Networks – Implementation Issues and Solutions," *Applied Research Lab, Ericsson Ireland*.
- [3] "What is Arduino?," [Online]. Available: <https://www.arduino.cc/en/Guide/Introduction>. [Accessed 14 July 2016].
- [4] "Arduino Board Uno," [Online]. Available: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. [Accessed 14 July 2016].
- [5] "ArduinoBoardMega2560," [Online]. Available: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>. [Accessed 14 7 2016].
- [6] Suyono and Sudarmadil, *Hidrologi Dasar*, Yogyakarta: Fakultas Geografi UGM, 1997.
- [7] A. Gaddam, M. A. Hrooby and W. F. Esmael, "Designing a Wireless Sensors Network for Monitoring and Predicting Droughts," *Wintec Research Archive*.
- [8] "Relative humidity," Lenntech BV, [Online]. Available: <http://www.lenntech.com/calculators/humidity/relative-humidity.htm>. [Accessed 27 June 2016].
- [9] R. M. Mehra, *The Healthy Office: Creating a Healthy Office Environment*, Excel Books, 2006.
- [10] "WHO Guidelines for Indoor Air Quality: Dampness and Mould," World Health Organization, 2009. [Online].

Available:

<http://www.ncbi.nlm.nih.gov/books/NBK143941/>. [Accessed 21 July 2016].

- [11] "*Relative Humidity* over the last 24 Hours (hourly data) for Vancouver," [Online]. Available: http://vancouver.weatherstats.ca/charts/relative_humidity-24hrs.html.

BIODATA PENULIS



Regin Iqbal Mareza, lahir di Sampit, 23 Juni 1995. Anak sulung dari 3 bersaudara. Sempat menempuh pendidikan di SDN 02 Sampit (2000-2001), SD Al Azhar 21 Pontianak (2001-2003), SDN Sukadamai 3 Bogor (2003-2006), SMP Negeri 1 Bogor (2006-2009), SMA Negeri 1 Bogor (2009-2012), dan S1 Teknik Informatika ITS (2012-2016).

Selama menjadi mahasiswa penulis aktif dalam organisasi mahasiswa Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) dan Keluarga Muslim Informatika (KMI). Menjadi staf Departemen Media Informasi HMTC (2013-2015), staf Departemen Kaderisasi KMI (2013-2014), dan diamanahi menjadi kepala Departemen Kaderisasi KMI (2014-2015). Penulis juga pernah menjadi juara dalam lomba kepenulisan tingkat Surabaya, dan juara I lomba menulis cerpen yang diadakan oleh CAKRA App.

Penulis dapat dihubungi melalui alamat surel reginiqbalmareza@gmail.com